The Australian Apple Review

The Independent Magazine

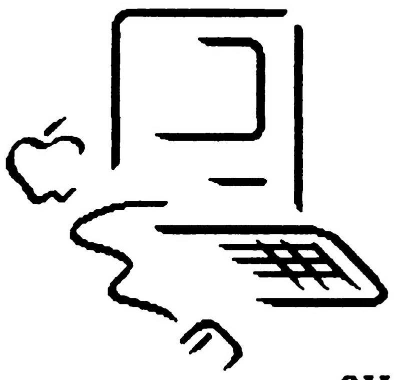
Vol 1 No 8 September $3*

A Gareth Powell Publication

Part III Frie to which us
ng BASIC "Step by Step" Logo Vs B
education Disk to tape dump utility
ing in "South Dakota" More disk storag
der DOS 3.3 Disk drives in the future Nu
er Farm/Rodeo/Circus A Pot-purri of App
elights Brian Hammerhead is the "Hellfi
Varrior" Worm in the Apple Editorial Let
s News Word processing at home: a profit
enterprise Repairing your Apple – Part
Friendly to which user? Learning BASI
p by Step" Logo Vs BASIC for educatio
to tape dump utility Farming in "S
ta" More disk storage under DOS
ves in the future Number Farm
A Pot-purri of Apple deligh
ead is the "Hellfire War
Letters

Registered by Australia Post Publication No. NBG6313

*Recommended retail price



CompuMusic

invites you to a preview of

™ **Macintosh™ Software**

available from all fine Apple® Dealers

™

FILEVISION™

Words, numbers, pictures : think & work in all 3.

VISUALIZE INTEGRATED:

Planning, Tracking, Scheduling, Locating, Deciding, Discovering

- put pictures and information together in a filing system
- draw lines, rectangles, squares, ovals, circles, and free-form shapes
- stretch, shrink, reshape, shade, cut, paste, move, and copy
- overlap, hide and highlight
- change and create symbols
- print lists, labels and pictures
- change print styles and sizes
- record, edit, and interrogate visual, numeric and written information

THE SALES EDGE™

Business Strategy Software

PERSONAL POWER WITH A STROKE OF GENIUS

- A guide to the business jungle with step-by-step strategies
- Manage people, not just words and numbers
- Evaluate human factors that affect your decisions
- Discover the best tactics to stack those factors in your favour
- Obtain a printed 3-10 page analysis of your business situation

PLUS

LANGUAGES	GRAPHICS	GAMES
• Macforth	• Mac the Knife	• Millionaire
• Aztec 'C' Compiler	• Cut 'N' Paste	• Macslots
	• Clickart	• Funpack

☎ (02) 692 9293 ☎ (02) 692 9875

The Australian Apple Review

The Australian Apple Review

The Independent Magazine

Vol 1 No 8 September $3

A Gareth Powell Publication

Part III Friendly to which us
ng BASIC "Step by Step" Logo Vs B
education Disk to tape dump utility
ing in "South Dakota" More disk storag
der DOS 3.3 Disk drives in the future Nu
er Farm/Rodeo/Circus A Pot-purri of App
lelights Brian Hammerhead is the "Hellfi
Varrior" Worm in the Apple Editorial Let
s News Word processing at home: a profit
enterprise Repairing your Apple – Part
Friendly to which user? Learning BASI
p by Step" Logo Vs BASIC for educatic
to tape dump utility Farming in "S
ta" More disk storage under DOS 3
ves in the future Number Farm
A Pot-purri of Apple deligh
ead is the "Hellfire War
Letter

The Australian Apple Review
Top rear, 4 Carrington Rd,
Randwick, 2031. N.S.W.
Phone: (02) 398-5111.

Published by Gareth Powell Pty. Ltd.

Editor: Gareth Powell
Assistant Editor: Leslie D'Bras

Art Director: Roy Bisson
Production Manager: Gary Williams
Production Assistant: Sian Powell
Photographer: Gary Williams
Subscription Manager: Tina Spathos
Phone: (02) 398-5111.

Advertising Manager:
Craig Hoskinson
Top rear, 4 Carrington Rd,
Randwick, 2031. N.S.W.
Phone: (02) 398-5111.

Typesetting:
Printaction Pty. Ltd.

Distribution:
NETWORK, Sydney.

Printed by:
Ian Liddell Commercial Printing.

Contents

EDITORIAL

The news that Apple had brought out the Macintosh with a vastly improved memory came as something of a surprise. No one in Australia expected the machine to be available until early next year.

With this advanced step forward Apple are securing the position of the Macintosh in no uncertain way.

It is already known that Apple expect that in Australia alone the Macintosh will next year account for 5Ø percent of their turnover. Which is an awful lot of Macintoshes.

Certainly it is making a big impact on the educational market, but what has surprised us is the way it is being snapped up by businessmen and computer hobbyists.

I must admit I was rather sceptical about the Macintosh and its chances of breaking into the big time when it was first launched. I no longer harbour such doubts.

Two elements seem to have been involved in making the Mac the major success that it is – apart from its technical excellence and superior performance.

The first is the problem that IBM are having with the machine they launched as the PCjr. It really does not matter what attempts they make to alter and improve this machine – the end result is always going to look like a muddle. And the American public, with immense good sense, are staying well away and, in many cases, buying the Macintosh instead.

The second development is the flood tide of programs that have been released. At the beginning it could have been thought that the Apple Macintosh programs were more in the minds of their PR companies than reality.

And at the beginning that may have been true. Now, however, we see new programs being released almost every day of the week. And as each program finds its niche, so the likelihood of the Macintosh becoming the pre-eminent computer in its range increases.

In fact, thinking about it, the

Macintosh doesn't fit into any range. It stands uniquely alone. The only valid comparisons would be with the Lisa.

Because of all this excitement about the Macintosh, starting with the next issue we will be running an Australian Macintosh supplement with each issue. We hope you will enjoy it. □

LETTERS

Dear Sir,

On behalf of my fellow members of the Brisbane Medfly User's Group, can I ask you to break the suspense? We have all been waiting eagerly for your article on the Medfly, the more so because of your dark hints about its nativity.

While we are inclined to agree with your comments about support from the manufacturer, the fact remains that Medfly is one of the few Apple compatible machines that is not a "fake", and is reasonably well constructed and documented. Furthermore, as far as I know, it is also the only one that is significantly more powerful, and actually has an implementation of CP/M 3.Ø available.

The fact that your's is one of the few magazines that doesn't pretend that compatibles don't exist has impressed us; I hope you will bear the fact in mind when contemplating policy. Software reviews which indicate Medfly compatibility, occasional articles specific to us would be welcomed. In addition, could I enter a plea that CP/M isn't forgotten?

Some of your correspondents may dislike it; but CP/M users with a 65Ø2 in their machine still constitute a very large proportion of the total. DOS software isn't all!

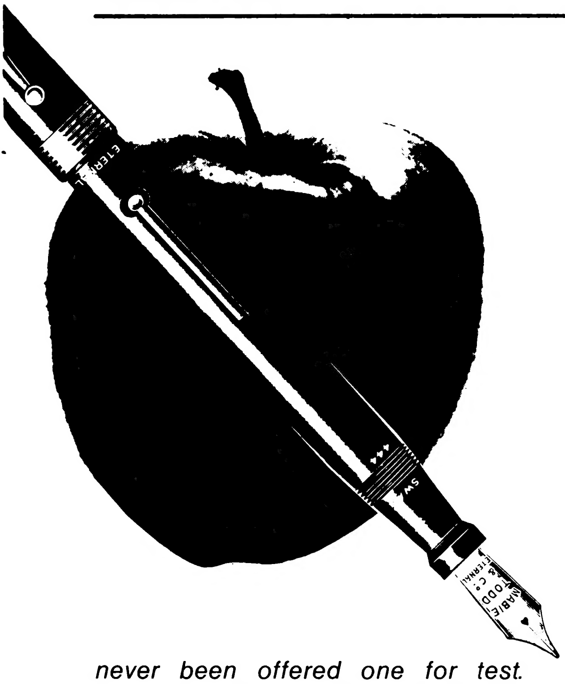
I would be grateful too, if you could include the name and address of the Brisbane Medfly Users Group in your next listing of user groups, (12Ø Highgate St, Coopers Plains, Qld 41Ø8, for those who are interested). The group has been operating on an informal basis for about a year now. Membership has risen to just under 2Ø, and is steadily growing. The group is in touch with overseas groups, including two in the USA, and is looking forward to being able to expand its contacts.

At present, we can offer members help in setting up and choosing software; for the future, we are hoping to establish a library of public-domain software for the Medfly.

Dr K J Walker
Coordinator
Coopers Plains, Qld

Ed: *A fair letter. As it happens, I flew back to Hong Kong with one of the people who are behind MedFly and he told me the machine is now being sold into the People's Republic of China.*

But your letter, in a sense, answers its own questions. The reason we have never reviewed the MedFly (although I have used one in Hong Kong) is because we have



never been offered one for test. Secondly, the number of MedFly users in Australia is miniscule compared to the number of Apple users (genuine and fake). Obviously we are catering for the majority of our readers. I have nothing against the MedFly (except for its damn silly name which is a schoolchild's pun referring to the invasion of the Mediterranean Fruit Fly into the apple orchards of California) but I must state its documentation is so bad as to be ludicrous and the dealer network in Australia, vestigial. Nope, I can't see us catering to the needs and desires of MedFly users until and unless someone shows me they are a significant part of the market. For MedFly users who do not want to remain out there on their own the offer from the Brisbane MedFly Users' Group will no doubt initiate some activity.

Dear Sir,

I have read with interest your article in the Weekend Australian on August 4-5 re advice on purchasing a computer.

My wife and I operate a small coffee shop employing four people.

My turnover is between $3,5ØØ to $4,5ØØ per week, all sales are cash over the counter, so we have no debtors.

We have approximately 35 suppliers, some cash, some account.

Could you please advise us as to a suitable computer and the necessary software and or other equipment that would be necessary to take out following information:

1. Total sales per week and month.
2. Total purchases as well as individual totals.
3. Percentages on purchases, sales, wages, etc and against rent, plus any other information you may feel is necessary, or any other information that you know is available that is relevant.

Whilst being human I do not wish to spend a lot of money, and do not know anything about computers. However I would be prepared to spend approximately $2,ØØØ if the above information could be supplied simply without the tedious job of manual additions etc.

Hoping you are able to assist. I enjoyed reading your article, even though I am a raw recruit and have never operated any computer.

Frank C. Koschel
North Mackay, Qld

Ed: *It is very easy to give glib answers when you do not have the full story. But from the information given an Apple IIe using a Cashbook produced by Zofarry with an el cheapo dot matrix printer would fill the bill. Cost a bit more than $2,ØØØ, though. Best bet is to buy a good second hand Apple with the same program.*

Mr Gareth Powell,

May I say how much I like your page. I am a freelance technical writer and I have been using an Apple II+ with Zardax as a word processor for three years. I am very satisfied with the combination.

There are several hundreds of hours of work on my Zardax files and I would like to upgrade to IBM, Lisa or Macintosh, but every enquiry I have made seems to indicate that if I wish to access this bank of work and bring it up for including in the book, I am stuck with the old system based on the 65Ø2 assembly language?

All I ask is that I can save retyping by transferring text to the new format. Everyone says it cannot be done. Cannot X3!!!? I am sure that many of your readers are in the same position but it seems to be a taboo subject for all computer journos. Please why?

Looks that I will have to go to Apple IIc if I want a new trick?

Some years ago as Mr Remington Rand in Australia, I introduced the portable typewriter to retail stores. Previously all typewriters were direct sales from the manufacturers. It was a mini PC story, suddenly portable typewriters as a homeware product boomed on HP for the first time. One observation was that the salesmen, and even the best, never seemed to lower themselves to try and learn how to type. It was accepted that if you bought one, learning to use it was nothing to do with them. Some actually could peck out "The quick brown fox . . . " on two fingers; big deal! Perhaps this attitude is repeating itself in the PC boom?

Malcolm Goldfinch
Woollahra, NSW

PS: Although I was a journalist before the war and for a while after, I somehow was the first person in commercial computers in Australia, lecturing on the only non-laboratory computer, Univac, to post graduate courses at the Adolf Baser Computer Laboratory at the University of Sydney in 1956.

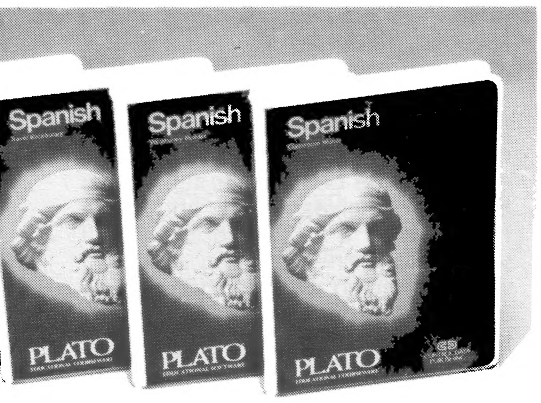
Ed: *With that wealth of experience, he is asking me? I cannot see what the problem is with the text files created by Zardax. A text file is a text file is a text file as Alice B. Toklas or Gertrude Stein (or possibly both) used to say. I transfer text files back and forth from all sorts of machines by pretending I am flashing them down the telephone line. For example, this limpid prose is being written on an Apple and will shortly appear on a Wang. To do this I transmit it through a communications package and a CCS board down through an RS 232 port and into the Wang. The Wang thinks the signals are coming in through its telecommunications program, the Apple thinks it is transmitting information down the telephone.*

This works nearly every time providing you are sending text files that have been written in ASCII, or something very close, which is true of almost every word processing program that I know.

You may have a few problems with line length so you get the occassional unwanted paragraph inserted but that is pretty easy to fix once the copy has been transmitted.

I do not know what X!!! means but it sounds rude to me.

Apple news



Control Data have been in computer education and computer publishing almost from the beginning. Now, in Australia, Control Data Publishing – a new branch of the company – has released a flood, a positive deluge, of programs for the Apple. These programs are all educational and appear under the generic title of "Courseware". They allow your Apple to take you through the rigours of a number of subjects such as Basic Numeric Facts, Spanish Travel, Spanish Shopping, Spanish Classroom, Polynomials, Building Relationships, Forming Positive Behaviour and Physics.

All of the titles are competently written and produced, albeit with an American slant. New titles are being originated in Australia which we should see in the coming year.

Victorian first

In August 1972, Burwood High School in Victoria was the first Victorian high school to have its own computer.

By August 1978, there were three Victorian high schools with their own computers.

By June 1984 all of the more than 32Ø high schools and more than 14Ø technical colleges in Victoria had their own computers. Most of them were Apples.

The Education Department in Victoria is, perhaps, the most computer oriented of all Australian education departments and has set up a joint software production house to make sure that the software is as good as the hardware.

An Apple for the teacher

Apple has provided a twenty machine Apple II family personal computer laboratory to assist in the training of teachers for the Association of Independent Schools.

The 2Ø computers have been installed in a special computer classroom at the northern Sydney girls school, Abbotsleigh. During school hours, the classroom will be used by the 1,ØØØ odd students, but is also expected to become a major community resource for northern Sydney suburbs. The installation is valued at more than $26,ØØØ.

More for the Lisa

Apple has introduced Lisa 7/7, an integrated software package for the Lisa personal computer than combines seven powerful and diverse business functions.

Lisa 7/7 includes project management, word processing, spread sheet, data communications, data base, business graphics, and structured graphics. It follows closely in the steps of Lotus 1-2-3, Symphony, Appleworks, Framework and many others. This package allows a buyer to cover most requirements with one software purchase.

The sensible move would be for Apple to put all of this software into a series of chips so that the Lisa comes totally configured and ready for action.

Our guess is that this will indeed happen in the early part of 1985.

For the moment the package is only available in software at a price of something under a thousand dollars. Which, considering the scope of the package, is a bargain indeed.

The painting mouse

Ahware (where do they think up these names?) of Danville, California have released a Mouseprint program which allows Apple's Mousepaint program to print directly on printers other than Apple's Dot Matrix or Imagewriter.

This is no great deal, as far as we can ascertain, because all that is happening is you are getting a graphics dump which is already an essential part of Apple's Mousepaint program. Versions of Mouseprint are available for most popular printer interface combinations.

Printers currently supported include the Epson, Okidata, C. Itoh and Star Micronics lines. Interfaces currently supported include Epson, Grappler, Pkaso, Prometheus, Microbuffer, Graphicard, Apple and Printerface.

Available at US$22.95 from Ahware, 8Ø5 Luz Court, Danville, CA 94526, USA. On the other hand you can save yourself the money, get out your printer's book of instructions and write a little driver program yourself.

Macintosh counts

A new line of accounting software designed specifically for the Apple Macintosh will be released by Interactive Applications in early November. The three packages are Accounts Receivable, General Ledger and Stock Control. Each is a stand alone system, and is written to function on a standard Macintosh with a single 4ØØ KB diskette. They make extensive use of the mouse, are written in Microsoft BASIC, and require little or no on site installation, training or support.

However, the amount of accounting you can do on the Macintosh until the new memory chips become available is fairly limited. OK if you are a small business or self employed. Not much cop for a full scale business.

Apple have recently introduced Pascal for the Macintosh, and it will be available in Australia in the next few months.

This is fairly logical as the major sales of the Macintosh todate have been to educational establishments and Pascal was originally designed as a learning language for students. Launching the Macintosh without Pascal might be regarded by harsher critics than us as a marketing hiccup.

The program is referred to as Instant Pascal, and is designed as a student's aid giving instant feedback.

The program checks each Pascal statement for proper syntax as it is entered, and displays any unrecognisable text in hollowed lettering. At any time, the users can ask Macintosh Pascal to check the entire program for consistency and proper syntax.

It is claimed Macintosh Pascal contains virtually all of the functions and standards of American National Standard Pascal and is compatible with Lisa Pascal.

But is it as fast as Turbo Pascal, we hear a voice ask. Let us move on to the next subject.

Chip checker

This is a double goody.

Australian Video Presentations have brought out a Chip Checker for Apples and Apple compatibles. Designed as a peripheral card, the chip checker checks, verifies and identifies all TTL IC's.

It is slot independent and is connected via a ribbon cable to a zero insertion force socket placed outside the computer. Used in conjunction with the software provided, the Chip Checker will test standard TTL (54/74ØØ series), low power Schottky TTL (54/74LS series), Schottky clamped TTL (54/74S series), low power TTL (54/74L series), high speed TTL (54/74H series) and TTL equivalent CMOS devices (54/74C series).

The Chip Checker not only tests for complete failure with the operation of an IC but by means of a loop routine will also test for chip instability and respond accordingly.

If a chip has been inadvertently scrubbed (or advertently – is there such a word? – scrubbed by a manufacturer who wants to hide what works where) and left with no means of identification, the Chip Checker will search and identify which type of IC it is. The Chip Checker has a recommended retail price of less than $3ØØ.

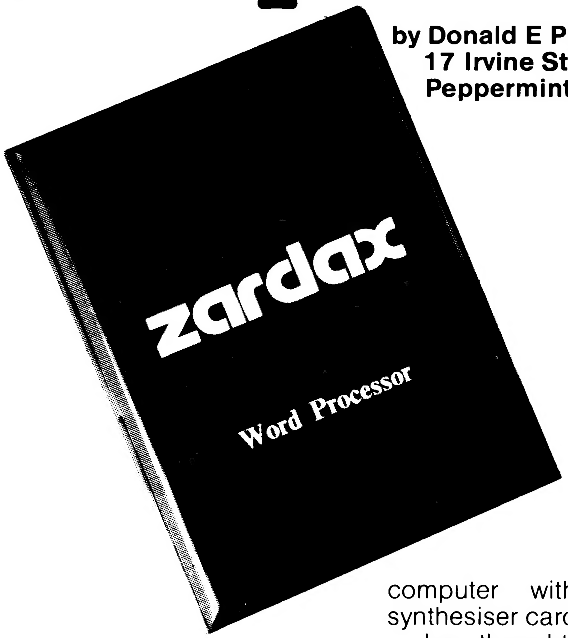
Actually, we have such a chip checker which we bought in Hong Kong. There are several versions available. Ours seems to work on the same basis as the Video Presentations board but it is not a copy.

These boards are amazingly useful if you have a problem which you have tracked down to being somewhere amongst those serried ranks of chips. It will test each chip and give you the good oil.

We can't see it being bought by every computer user but it should find a ready sale in schools, and with happy home hackers like us. □

Word processing at home: a profitable enterprise

by Donald E Pugh
17 Irvine St
Peppermint Grove, WA 6011



Each year thousands of young students leave school to find themselves on the dole. For those with typing ability, an Apple computer may seem to offer the key to a lucrative cottage industry; word processing. However, the business is not as easy as it sounds. Here is the experience of one person who took advantage of this cottage industry and now earns a respectable income with an Apple II+.

Joe Blake of Perth, Western Australia, spent sixteen years in the Army with his last nine years in the psychology section. One day in 1979 Joe experienced his first computer. "It was a pivotal point in my life," Joe recalls. "I was out for a short stroll and passed a small handwritten sign at the city hall saying 'computer display inside'. Walking inside, the first thing I heard was the Flight of the Bumble Bee played on an Apple computer with an ALF music synthesiser card."

Joe thought "Wow, that's really something." Next to the Apple was a Tandy computer that was talking. Joe, talented folk musician, meditated: "Which do I want, a computer that talks or a computer that plays music?". Joe debated the problem for five minutes, then asked the Apple dealer "How much?" A week later Joe owned an Apple computer system.

'An Apple computer may seem to offer the key to a lucrative cottage industry'

Joe entered his future typing career in 198Ø. One of his friends at the local Brisbane Apple Users Club was Ian Phillips, the author of the Zardax word processor. Ian first introduced the idea of word processing to Joe. A friend of Joe's was also learning typing and launched a career typing theses at the local university. Following a discharge from the Army, Joe was also looking for an alternative job. He had found a low rental house in Perth and already possessed the Apple. Because he read avidly in a variety of university fields and was a good speller, word processing for university students seemed an ideal career.

Following an extensive market survey, he purchased a NEC Spinterm printer for $4,000. He also purchased Zardax, from his friend Ian Phillips. Joe commenced working with this program and gradually developed real competence after numerous phone calls back and forth to Ian in Brisbane.

Joe's first typing contract was with the previous tenant of his house. Joe had mentioned he was typing theses and essays. "Before I knew it," Joe states, "this fellow who was majoring in psychology employed me to do all his typing during the next four years."

Joe thought it would take at least a year before he became known and to build up a clientele. It has probably taken two to three years. Joe's main advertising is by word of mouth, although he runs a commercial advertisement in the Yellow Pages in the typing section. He also advertises at the University Guild as an experienced typist and hands around his business cards freely. But he notes: "If it is possible to get a couple of really good customers who are convinced of your work, then business keeps flowing in second, third, fourth and even fifth hand. One cannot buy this sort of recommendation for love or money."

Joe has now been typing theses for the university for a period of four years. He finds: "I start from the beginning of July getting enquiries for theses. By August a thesis might come in. By September/October I'm overloaded with work and turning it away to two or three other typists. This work load continues up to Christmas. The slack period is during the Christmas holidays and the autumn period."

Joe finds that fifteen to twenty page essays are his real money makers. He gets immediate payment, while typing a thesis may delay his income for several months.

Joe attributes his success to excellent quality and low prices. He recommends that high quality typewriter output is essential, particularly for theses. He advises: "It's necessary to read anything that one can lay one's hands on. I type essays on dramatic art, chemistry, lobsters, botany reports, ecological impact studies – you name it, I've done it. I often interpret poor handwriting and it helps to have a background in the subject." Consequently it is essential to read broadly. Joe subscribes to "Scientific American" and similar magazines.

Joe also recommends that a basic understanding of Latin is invaluable for typing for doctors and for botanists. It assists in making informed guesses at the meaning of various terms. With the word processor, of course, it is always possible to put a question mark after the word and to correct it at a later date. It is helpful, though to be able to guess the word from the context. This saves time in corrections and makes it possible to return a piece of work with hardly any errors.

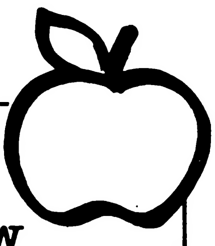
Joe is very pleased with the tumbling prices for Apple equipment. Also the price of quality daisywheel typewriters has really dropped. This has made entry into the word processing field easier for others. Yet Joe still finds there are thousands of people around with typing expertise, hundreds of people with computer expertise but few who can make these two skills come together to produce a quality result.

He also finds it useful to have some skills in computer repair and equipment maintenance. Joe is able to replace chips on his computer and regularly winds his ribbons for the NEC Spinterm printer.

Joe has recently purchased a MedFly, an Apple compatible computer. He now rents his older Apple and office space to another typist. They charge by the keystroke (business, one cent every five strokes, students, one cent every seven strokes), and find they cannot keep up with the business.

The secrets of running a home word processing business and of his success, Joe concludes, are: low typing costs, quality equipment, low overheads, and provision of error free competent service. Joe clearly provides all these services in a friendly way. □

Joe Blake, Proword Enterprises
(09) 381-1980



There really is a difference in diskette brands.

It's in the way they are made.

Because the computer industry cannot afford variation in the quality of diskettes, at Nashua we looked for, and found a way to ensure absolute consistency.

Here's how we do it.

Quality Circles.

At Nashua we've found the best way to attain this 'consistency' is to ensure that at each stage of production our diskettes are statistically checked to make sure the quality is

'built-in' every step of the way.

Rather than long production lines, we have 'Quality Circles' – small groups of people whose job it is to make sure that each Nashua diskette is right in the first place.

The result is a diskette with such consistency, that it is chosen by those people who can't afford mistakes.

Phone Sydney 958 2044, Canberra 47 0511, Melbourne 428 0501, Brisbane 369 4244, Adelaide 42 0021, Perth 328 1888, Darwin 81 6204, Auckland 77 5468, Wellington 85 1965, Christchurch/Dunedin Whitcoulls Ltd, PNG 25 6766.

Nashua

There really is a difference.

Ruwald & Skinner 402 NA/A

Repairing your Apple-Part II

Once you have done the simplistic diagnostic trouble shooting we suggested in our last issue, you have to move on to slightly more complicated matters. The amount of equipment you need is absolutely minimal. What you do need most of all is organisation. You simply cannot have enough little boxes in which to put screws, IC chips and bits and pieces. Once you get them mixed up you will have the devil's own job to start off with. When you are going to repair your Apple the first thing you need is a big soft surface. A large towel laid flat on a desk will be ideal. The second thing you need is a lot of light. It depends on your age and the strength of your eyesight but I find the minimum is one really good desk light and a powerful torch. You may need less if you have the eyes of a young Augus.

You will need a pencil and paper to make notes and sketches of everything you take out and where it fits. You will also need a Phillips head screwdriver in order to remove some of the screws at the bottom of the machine. As well, you will need a small flat bladed screwdriver to deal with some of the non-Phillips screws.

Most professionals prefer to yank out the ICs using a screwdriver. This way can lie disaster. It is much easier for you to use an IC puller which costs almost nothing. They are available at Tandy's for about $3.

Now let us presume your initial run through the machine as listed in the last issue has not worked. That is, you have removed all the boards and tested to see if the machine will start. Then you have to take your Apple apart. It really is not that difficult. First of all approach it step by step.

Step one:

Turn off the power at the computer.

'If you use brute force and ignorance you bend the pins of the chip'

Open the lid and touch the power supply in the left-hand corner (it is the big gold or silver box) with your bare hand. You will not feel any shock but it will automatically earth any static electricity in your body.

Step three:

Remove the socket from the mains. Then to be absolutely sure, in our old mother worrying about the child damaging himself mode, pull the power plug from out of the back of the computer. This means your computer cannot possibly give you a shock, which is a bad thing, or you pull out boards when there is power going through the computer. Which in our opinion is even worse. Start removing the peripheral boards one by one. As you remove them write down on a piece of paper where they were and the direction in which they were pointing. Most peripheral boards appear to be fairly logical so the long end faces towards the keyboard and the short end with the contacts towards the back of the machine. But not all. There is a printer interface made in New Zealand which is almost exactly the reverse. So you have to check them with great care. Take out your peripheral cards one by one, starting with slot Ø, write down what they are and the direction in which they were facing, and lay them very carefully and gently down on a large piece of soft cloth in their correct order and facing in their right direction.

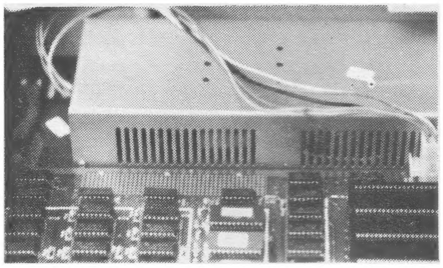
It is possible that the order will be:

Slot Ø a 16K RAM add on board
Slot 1 a printer interface
Slot 3 an 8Ø column board
Slot 6 a disk drive.

You may have many more boards than that. You may have fewer. Disconnect them and take them out very carefully. You should now have no wires attached to your machine whatsoever. Check that this is so.

Step four:

Turn the machine up-side-down so it is resting on the soft towel surface on the desk. Start to remove the Phillips head screws from your machine. If you have a real Apple



there will be ten Phillips head screws to remove. If you have a fake Apple you may find you have a lot fewer. Most fake Apples come with a shortage of screws.

The first two screws to remove are at the back edge of the bottom plate. That is the edge furthest away from the keyboard and these two screws are on each side behind the rubber plate on the extension. Note most carefully there are several other screws in line with the rubber feet. In fact there are screws all around the centre of the base board. Do not remove these or you will spend a considerable time trying to sort it all out again. Remove only the two screws located behind the rubber feet.

As you remove the screws do it most carefully with a steady pressure. Make sure you have a good condition Phillips screwdriver. (There are others called Pozidrive which may not fit exactly.) You want the perfect match between screwdriver and screw. Then ease those two screws out. Once they are out put them carefully into one of your containers. Then along the edge on each side of the machine are two screws. These four screws in all, are removed next and placed in the same container. Finally there are four Phillips head screws along the front edge of the bottom plate and these are located in semicircular notches. When you take these out, you will almost certainly find they have a lock nut (a sort of serrated edged washer) fitted to them. Make sure you do not lose them. Now you should have ten screws set to one side in a cup and your machine is ready to be taken apart.

Now with great care grasp the whole of the Apple housing and the bottom plate with both hands and keep the two sections firmly clamped together as you turn the computer face up again. Do not let it slip and do not let the two parts come undone. Put your hands inside the housing, grasping the housing close to the keyboard. As you lift up you will see a space of a few centimetres and inside that space you will see a coloured ribbon cable which goes from the keyboard to the mother board. It consists of a multitude of cables stuck together and is about three centimetres wide and has an absolutely flat appearance. It looks more like a ribbon than a cable. Note carefully how this cable is attached to the mother board.

'A large towel laid flat on a desk will be ideal'

This flat cable is attached to the mother board by an IC. It will need to be removed extremely carefully. Especially the first time you do it. It is almost impossible to get an IC puller in to take out this particular chip. Therefore you are going to have to insert the flat bladed screwdriver from the front end of the keyboard through the gap you have created by lifting the body. Now insert it between the socket and the IC and twist the screwdriver very slightly sideways and the IC will just start to lift. Rake out the screwdriver and lower the front of the machine. Now working from inside the computer and using a torch put the flat end of the screwdriver between the chip and the socket again and make a half turn. The chip will lift out of the socket at that end. Repeat the process two or three times until the chip comes out very easily.

Notice that if you use brute force and ignorance you bend the pins of the chip. You can probably straighten pins on chips about three times before they snap off but do not bet on it. And once you have bent them they are extremely difficult to get back into their sockets. So it is best to approach it very slowly and very carefully until you have got that chip removed from its socket. With your pencil and paper note carefully the way that the ribbon aligns with the chip before it goes into its socket. Otherwise, like many other home repairers (including myself) as sure as god made little Apples you are one day going to try and put it in the wrong way around. And the keyboard will not appreciate it. You can now lift the whole of the top of the Apple away from the base leaving the mother board in all its naked beauty.

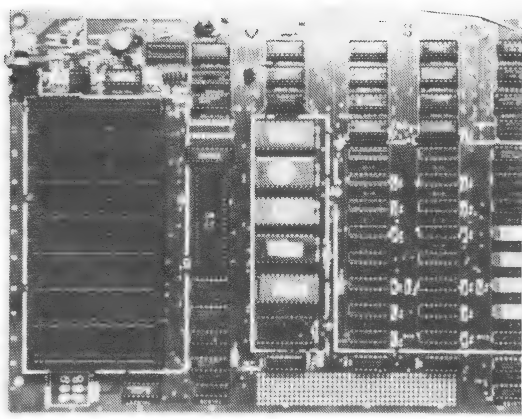
The mother board is mapped out just like a city and you can see intersections, maps and numbers printed on it in white ink. The numbers 1 to 14 are printed in white letters across the front of the board. Using this extremely simple method of identification you can find your way to any part of the board. For example C2 is an IC which has a number printed underneath it. That number is 74S195 IC. Easy to find. You follow an imaginary line across the board from left to right at location C. Count along 2 and there is the IC.

This is where I diverge from all other people recommending working on an Apple. It seems to be general practice at this stage to teach people to be able to remove the mother board from the base plate. I cannot honestly think of one good reason why you should ever want to do so. If the computer is so damaged that the mother board has to come away from the base board then it is not a job for the happy home hacker. It is a job for a serious professional. And in all fairness, in all the time I have been working on Apples I have never seen a situation where I have had to lift the mother board away except where I have been replacing a very indifferent piece of Hong Kong and Taiwanese soldering with a superior article. That should not happen to you.

'Then you have to take your Apple apart'

You will notice on the right hand side there is a two prong plug with a white plug on it with two wires leading to a mickey mouse loudspeaker on the other side of the computer. These rarely, if ever, go wrong. If they do, they can be replaced for a few dollars. It may be when you do, you

The Apple motherboard



would like to get yourself something better in the line of speakers. On the other hand you may find if you do, the noise will be deafening. With me it is an invariable practice to change the loud speaker and put a volume control on so I can have it at full chat when I am playing Star Wars and extremely quiet when I am playing with my word processor. It is all a matter of taste.

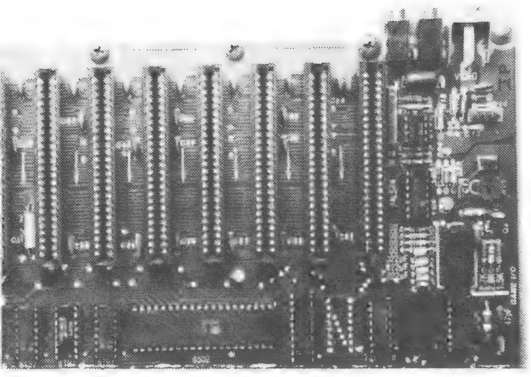
If you are going to play around with the speaker, bear in mind it is glued in. It is relatively easy to lever it up with your straight screwdriver and glue it back again with rubber or super glue.

'This means your computer cannot possibly give you a shock'

Now we come to problem shooting.

The first major problem you will enounter can be with the power supply. However, on real Apples it is rare for them to go wrong. If you have been switching on your computer and seeing the power light come on, then you have very close to a 1ØØ% chance that the power supply is operating and operating well.

If however, you have a fake computer, then the first area to suspect is always the power supply. They are notorious for failing. The reasons for this are very simple. In the area in which they are built, the electricity supply (although it is a nominal 22Ø volts) is frequently extremely sub standard and fluctuates around the 18Ø volt mark. When these power supplies meet a full-blooded Australian current of 24Ø volts the


The eight card slots at the rear of the Apple motherboard

capacitors build up power and in theory the fuse built inside the power supply should go. Sadly, most power supplies from overseas do not have a fuse and the only way in which you can deal with this situation is to replace the power supply.

This next section applies only to people who are repairing fake Apples.

'You should now have no wires attached to your machine whatsoever'

Turn the machine over again and you will see there are four screws in the top right-hand corner which are holding on the power supply. Sometimes they are silver. Sometimes they are enameled, round-headed Phillips screws. Remove them and put them in a separate container. Now slip your hands underneath and clamp the power supply to the board and turn it back up. The power supply takes in the nominal 240 volt alternating current and sends it out in a very strictly controlled plus five volts minus five volts and plus 12 volts minus 12 volts and a negative signal. It then goes through a plastic connector onto the mother board. Now before you rush out and buy another power supply from the dealer who supplied the fake Apple in the first place, note well that these connectors vary widely. They may look exactly the same but very frequently they are not.

To disconnect the power supply from the mother board squeeze the sides of the semitransparent white connector which is towards the top end of the power supply. Squeeze these two sides together and the plug can be removed relatively easily. Note carefully that there are patterns and indentations on this plug to make sure that only the right male part fits into the correct female part.

Now it is perfectly true you can get almost any type of fitting from an electronic parts shop that will fit into the part on your board. We have done it many times. But it is odds against that you will find a power supply with the fitting already correctly set up that will fit precisely in the socket that you have now remaining empty on your mother board. The odds are about four to one against.

'The damn things are rivetted together'

There are two ways to get around this. Either, when you go to the shop you make absolutely certain that the fitting attached to the power supply they are selling you exactly corresponds with the socket on your board. To do this do a very careful pencil drawing with measurements. If no such power supply is available anywhere – and that is frequently the case – then you will have to take a standard power supply and refit it with a new plug that matches the socket on your power board. This is not an impossible task but it is time consuming and it does mean that the lead on the power supply will of necessity get shorter. Because the plugs are so fitted that they can not easily be removed without cutting the wires. I have tried every way known to man to remove those wires in tact including melting down the sockets. None of them worked. In every case I have had to cut the wire as close as possible to the socket and start again. If you are going to do this it is imperative before you start that you make a very precise diagram of which colour wires go into which holes. The kit that you can buy for the plug is fitted with little one-way metal grippers for the end of each wire. In theory you do not need to use any solder on the ends of the wire. In practice I always do because it makes it much easier to insert the damn things correctly. It depends how manually dexterous you are. If you are as clumsy as I am a drop of solder sweated into the end of the wires makes life a lot easier.

Try to dissect a fake Apple power supply to repair it'

This complication does not – repeat not – apply to owners of genuine Apples. This is because, as far as I have ever been able to ascertain, there is only one type of power connector with genuine Apple IIs and that connector rarely needs to be undone as the power supply (with the exception of the switch) appears to be relatively foolproof.

To repeat. If you have a fake Apple and the power supply goes wrong the most probable area for problems is going to be matching the socket on the replacement power supply with the socket on the mother board. Just because they look the same on the outside does not mean to say that they are going to be the same on the inside.

It is almost totally pointless to try to dissect a fake Apple power supply to repair it. Very rarely are they fitted with fuses. Unless you are an extremely competent electronic engineer you will not be able to trace where the fault is. And when you find it you will find that some of the circuit board has been damaged. It is cheaper, quicker and better to buy a new supply.

With Apple computers' power supplies even if they go wrong you can not fix them. The damn things are rivetted together. You may think that this is a diabolical liberty. So do I. Because within the Apple power supply there is a fuse and frequently this fuse blows. If like me you are impatient with people in authority who rivet things together instead of using screws you may, as I did, drill out all of the rivets in order to be able to replace the fuse. Only do that if your computer is out of warranty. Because if you do it when your computer is in warranty then Apple will take the view that your warranty is null and void and will charge like a wounded bull to fix the damage. (It is worth cogitating at the moment on the mind of the engineer who thought of the idea of putting a fuse within a power supply so that the power supply fuse can be changed and then rivetting the whole power supply together so that the fuse cannot be changed. There seems to be a contradiction there somewhere.) □

Friendly to which user?

by Ian Joseph

Computer-aided instruction is an idea whose time has come. But it must be user-friendly. This, however, raises the question:

```
10   REM  ***********************
20   REM  *    LISTING NO. 1     *
30   REM  *    Y/N     ANSWER    *
40   REM  *    ERROR      TRAP   *
50   REM  *          BY          *
60   REM  *      IAN JOSEPH      *
70   REM  ***********************
1000  HOME : VTAB (5): HTAB (3): PRINT
        "DO YOU WISH TO CONTINUE THI
        S TEST?": GOSUB 5000
1010  IF ANS$ = "Y" THEN 10
1020  HOME : END
1030  REM
5000  REM  SUBROUTINE TO DETERMIN
        E A Y(ES) OR N(O) ANSWER TO
        A QUESTION
5010  VTAB (10): HTAB (10): PRINT
        "ANSWER Y(ES) OR N(O)"
5020  GET ANS$
5030  IF ANS$ <  > "Y" AND ANS$ <
        `> "N" THEN 5020
5040  RETURN
```

User-friendly is a term one encounters often in what is no doubt the start of the era of a computer (or its derivatives –Beltel, video games etc) in every home. One constantly hears this phrase in connection with one or another program, but what does it mean to us amateur programmers hopefully writing programs with the intention of others using them eventually?

One definition I heard from an importer of software (he shall remain nameless) was: "Ahah, user-friendly. Does it ask you your name and all that?"

While having this feature in a program may make the machine seem more friendly, less threatening and less mysterious, it still does not make it any more friendly if answering the above question with a numeric input instead of with a name (string input) causes the program to give all sorts of mysterious messages which a non-programmer finds as difficult to understand as Egyptian hieroglyphics. By this I refer to error messages of the following kind:
?TYPE MISMATCH ERROR
?STRING TOO LONG ERROR
etc.

Obviously these make sense to other programmers, but to the average person they don't mean a thing.

In this article I shall attempt to explain what I mean by user-friendly and shall present a few subroutines which I hope will encourage you to go out there and write programs which your maiden aunt of 75 can use without any cries of distress.

My personal definition of user-friendly is – a program that does these two things: it will not allow the user of that program to make errors of the sort listed above. That is, if the user enters data in a form unacceptable to the program, the program will not crash but will tell the user that that particular form is not acceptable but another, of which an example is shown, is acceptable. Even more simply, it will not accept the input but will wait for the correct input to be given to it. For example see listing number one where a subroutine is given for trapping errors when a simple yes or no (Y(ES) or(N(O)) answer is required from the user. It will stop the user from entering extraneous information and possible error-causing information into the computer. For this example see listing number two, where, instead of asking a user which town is the capital of New Zealand and leaving it at that, the user is given a choice of numbered answers to press.

Finally, when programming one should try to limit the end user's use of the keyboard as far as possible as, by so doing, one lessens the chances of the user causing errors.

I suggest that you now switch on your computer and enter the program (Listing No 1).

Let us look very carefully at what each line does in this program:

1Ø-7Ø	REM statements
1ØØØ	This line first clears the screen with the statement HOME. It then positions the printing in the centre of the screen and five lines down from the top by using VTAB and HTAB. It now prints out our question to which a Y(ES) or N(O) answer is required and finally, using a GOSUB, goes to line 5ØØØ.
5ØØØ	A REM statement.
5Ø1Ø	The printing is once again centre positioned and printed five lines below the original question. The user is here told to answer Y(ES) or N(O).
5Ø2Ø	Here the GET statement is used to input information.
5Ø3Ø	If the answer is none of the

required ones, in this case Y or N, then the program goes back to line 5020 and waits once again for a key to be pressed.

5040 The program is told to RETURN and returns to the line after 1000, in this case 1010.

1010 At this stage having obtained an answer only in the form required we can continue with the program using this information in any way required.

A few notes about the program and some of its quirks. I personally prefer the form Y(ES) or N(O) to the form Y/N as the former tells us more than the latter. There is no ambiguity as to what the Y and the N represent. I also prefer using it as a subroutine because it is rare that one will find only one question in a program requiring a Y(ES) or N(O) answer. I also prefer inputting information by using GET than using INPUT as the user can press only one key at a time and, therefore, can make only one mistake at a time. The final point is the positioning of the printing in order to make it easier to read and understand. One need not necessarily be limited to Y(ES) or N(O) but can take into account other possibilities, ie M(AYBE) or S(OMETIMES) or A(LWAYS) etc.

A few words of explanation of Listing No. 2:

10-70 REM statements.

1000 HOME in order to clear the screen.

1010 VTAB and HTAB in order to position the printing. The question is now asked.

1020-1050 Possible answers are given and they are also neatly printed using VTAB and HTAB.

1060 The correct answer is given (No.3).

1070 The number of possible answers is given (four).

1080 The program is sent to the answer subroutine at line 5000.

5000 REM statement.

5010 Here the user is asked to choose an answer from (1) to AN, which is the total amount of possible answers.

```
10   REM  **********************
20   REM  *    LISTING NO.2    *
30   REM  *    ERROR   TRAP    *
40   REM  *    FOR   ANSWER    *
50   REM  *    ROUTINE  FOR    *
60   REM  *      NUMBERS       *
70   REM  *        BY          *
80   REM  *     IAN JOSEPH     *
90   REM  **********************
1000  HOME
1010  VTAB (5): HTAB (3): PRINT "
      WHAT IS THE CAPITAL OF NEW Z
      EALAND": PRINT
1020  HTAB (5): PRINT "(1) NEW PL
      YMOTH": PRINT
1030  HTAB (5): PRINT "(2) AUCKLA
      ND": PRINT
1040  HTAB (5): PRINT "(3) WELLIN
      GTON": PRINT
1050  HTAB (5): PRINT "(4) CHRIST
      CHURCH": PRINT
1060 RT = 3
1070 AN = 4
1080  GOSUB 5000
1090  HOME : END
5000  REM  ANSWER ROUTINE
5010  VTAB (20): HTAB (5): PRINT
      "CHOOSE AN ANSWER FROM (1) T
      O (";AN;")"
5020  GET ANS$
5030  IF  VAL (ANS$) > 4 OR  VAL
      (ANS$) < 1 THEN 5020
5040  IF  VAL (ANS$) <  > RT THEN
     5060
5050  GOSUB 6000: RETURN
5060  VTAB (20): PRINT "

        ": REM  FORTY SPACES
5070  VTAB (18): HTAB (12): PRINT
     "NO THAT IS WRONG": PRINT : HTAB
     (7): PRINT "THE CORRECT ANSW
     ER IS(";RT;")"
5080  GOSUB 7000
5090  RETURN
6000  REM
6010  REM  THIS CAN BE ANYTHING,
     A TUNE OR A MESSAGE TO REWAR
     D A CORRECT ANSWER
6020  VTAB (20): PRINT "

        ": REM  FORTY SPACES
6030  VTAB (20): HTAB (12): PRINT
     "THAT IS CORRECT"
6040  GOSUB 7000
6050  RETURN
7000  REM
7010  REM  SUBROUTINE TO WAIT FOR
     ANY KEY TO BE PRESSED BEFOR
     E CONTINUING
7020  INVERSE : VTAB (22): HTAB (
     7): PRINT "PRESS ANY KEY TO
     CONTINUE": NORMAL
7030  GET WT$
7040  RETURN
```

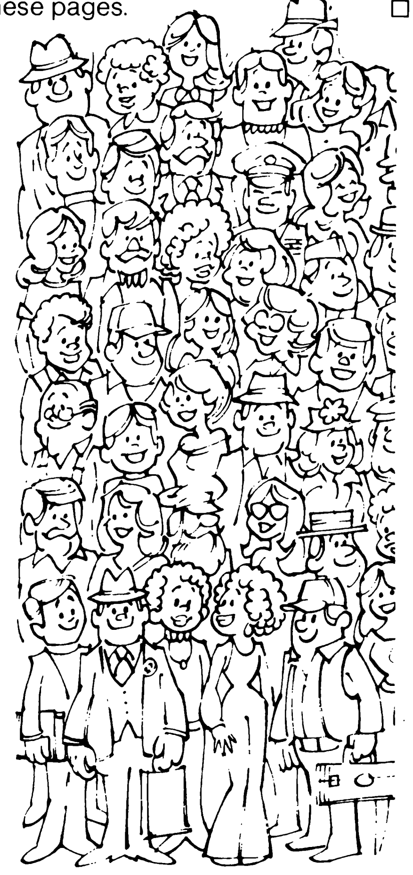
5020	GET statement is used to input information.
5030	The input is checked to see if it is an acceptable input and not outside the parameters. If it is unacceptable the program returns to line 5020 and waits for a key to be pressed.
5040	The answer is checked to see if it is correct. If it is not correct the program goes to line 5060.
5060-5070	The correct answer is given.
5080	The program goes to a subroutine at line 7000 and subsequently returns to the next question.
5050	At this stage the program is sent to line 6000.
6000	REM statement.
6010	REM statement.
6020-6030	Here the user is told that he is right and is given some sort of positive reinforcement.
6040	The program goes to a subroutine at line 7000 and subsequently the program returns to the next question.
7000	REM statement.
7010	REM statement.
7020	The user is asked to press any key (inverse writing).
7030	A GET statement waits for any key to be pressed.
7040	The program returns whence it came.

A few notes about the listing. Firstly, the user is not asked to type in the complete answer (Wellington). S/he is simply asked to press one key in order to answer (3). If required to type in the whole answer it is far too easy to make a typing mistake and type anything but Wellington, eg wellington, Wllenington etc. Secondly, the correct answer is at some stage given, not leaving the user to flounder if he doesn't know the correct answer. Thirdly, by using the wait routine in lines 7000-7040 (not to be confused with the BASIC command WAIT) the user can proceed at his/her own pace. Also, once again and most important, input is in the form of a string and only one at a time, thus ruling out the possibility of the program crashing if a letter is entered inadvertently.

There is of course an alternative to listing two and this is listing one in a changed form, ie ANSWER N(EW PLYMOUTH) OR A(UCKLAND) OR W(ELLINGTON) OR C(HRISTCHURCH). This is acceptable but, to my mind, not as neat as listing two.

The program given in listing two is meant to be the basis of a flexible multiple choice test and can be easily expanded to as many questions as you require. I am sure that, after having entered and tried these programs, you will easily get the hang of it.

There obviously are thousands more subroutines of this kind which I cannot for lack of space list here. I am sure that there are readers who have written routines taking these problems into account and I would really like to see them appearing in these pages. □



Learning BASIC "Step by Step II"

One of the best packages for learning BASIC is Step by Step. Originally we tested the very first Step by Step and found it excellent. Now they have Step by Step II and it is produced by PDI. In a series of five lessons it takes you through most of the basics of BASIC and also introduces you to some of the tricks of the programmers.

If you want to write relatively sophisticated programs then you need to go through Step by Step II right to the very end. If you do, at the end of it you will be a BASIC programmer. No doubt about it. The course materials consist of five audio cassette tapes, two program diskettes and a 96-page work book. All of these three elements are essential. In the original program we had, the diskettes were missing. With them you find your memory is stimulated visually and audibly. Each of the three elements supports and reinforces the subject of each lesson.

In the American style they give you an overview of what you are going to be learning and then present you with a specific fragment of that larger body of information. As each step is mastered you will find yourself being able to add it to a previous step, so you end up with a complete understanding of the subject. Each of the five lessons is broken down into two sections and each section is presented on a combination of computer displays produced by the program diskette and audio instructions from the cassette tapes. It works on the American method of multiple choice selection questions and at the end you will have learned the lesson through incessant repetition, none of it, strangely enough, being boring. After each mini lesson you are tested and then tested again to see that you are paying attention. We think the program could be slightly improved by administering a sharp short electric shock to anybody who is not getting the right answers, but the manufacturers have decided against this positive improvement.

The work book presents a summary of the lesson material and also has practice problems and programs that underline, reinforce, expand and amplify what has been taught. Typically, this includes answering several fill-in the blank type questions and writing small, five to ten line, programs.

All of us who have ever worked with BASIC will agree that inputting any program is a pain but the only way you are going to learn is to input programs. And then answer questions concerning the program after you have done it.

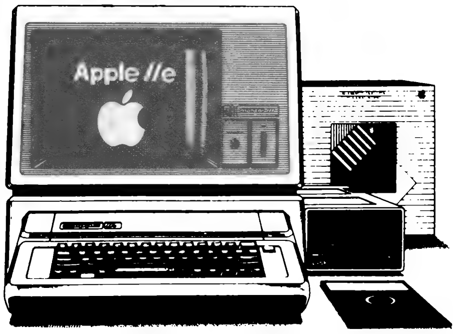
It is only after you have finished the first part of the lesson and completed the work book assignments that you can progress to the second part of the lesson, which has more practice problems, and then on to the lesson quiz.

Each lesson quiz is given and scored by the computer. Like the work book practice problems there are several multiple choices. There are also completion questions that will show if you have adequately mastered the lesson material. By the time you have got to this point, you have learned the lesson material visually, verbally and from the work book and you have been tested in at least three different ways.

If you do not score 8Ø percent on each quiz we are ashamed of you and we think you should sit down and think very carefully as to whether computing is going to be your bag. A child of 12 could walk through it. The publisher of this magazine managed to scrape through. Step by Step II is an excellent intermediate BASIC programming teaching system. It is considerably less expensive than going to a school and the system is designed to work with you on a one-to-one basis. Therefore, if like the publisher, you are extremely slow and peasant-like in your learning processes the program will stick with you and follow your speed. You will never be embarrassed because there is some smartass in the class who is galloping ahead at three times the speed.

The big disadvantage in using a system like Step by Step II is if you get stuck you do not have a teacher to ask. On the other hand you can get around this by joining any of the user groups, where you will be offered advice (quite a lot of it correct) until it comes out of your ears on any problem you care to mention.

This is undoubtedly one of the best BASIC educational packages available and one we can recommend absolutely wholeheartedly. □



Logo Vs BASIC for education

Computers and education

What can you do with a personal or microcomputer in the field of education?

The most obvious application is to run educational software packages developed for computer aided learning. These range from unimaginative flash card or simple question-and-answer type offerings to fairly imaginative exercises which allow the student to explore, discover and learn about the subject. Examples of the latter are some of the programs offered under the PLATO series by Control Data. Of course, the suitability of computer aided education depends to a large extent on the subject matter.

Another use for computers in schools is to familiarise the student with the operations and limitations of the computer. Such exercises can certainly foster more than a nodding acquaintance with common classes of software such as databases, word processing and graphics packages and electronic spreadsheets or integrated packages that combine the above features and then some. The student can then be better prepared to enter the job market or further vocational training requiring contact with computers.

The third use of computers in education, and one that is often touted in advertisements to sell computers, is to learn programming and to give Bruce or Beryl a head start in tomorrow's computer world. The question that bothers me most here is what should Bruce or Beryl compute? And which of the many programming languages should they learn?

BASIC vs Logo

For microcomputers, BASIC is one of the most readily available programming languages. One of its most attractive features is that it is interpretive. That is to say, the program is written in a human understandable form and the program instructions can be executed immediately – the computer interprets the instructions on the fly. For other languages such as FORTRAN or Pascal, the program instructions have to be converted (or compiled, in computerese) into machine code before execution. By eliminating this intermediate compilation step, there are considerable savings in overhead in computer memory and extra software, (a special program is needed to do the conversion to machine code) and in the turnaround time between making changes to a program and seeing the effects of the changes. This time saving more than offsets the loss in speed in the actual running of the program, especially in simple programs.

It has also been claimed that BASIC is easy to learn – because there are few RESERVED WORDS or PRIMATIVES such as FOR . . . NEXT, also the MAJOR limitations of BASIC. The very fact there are syntactical structures means that it is very easy to do simple tasks with BASIC but rather difficult to do something interesting. It is analogous to trying to write interesting literature equipped only with an early primary school vocabulary.

Indeed the "simplicity" of BASIC, which is often the reason given for promoting BASIC, arises not for the convenience of the user of BASIC. The limited expressive power of BASIC is really to make life easy for the computer designer – to make the process of interpreting and executing the human understandable instructions an easier task. For instance, the abominable necessity to append a "$" to a string variable or a "%" to an integer variable is really to let the language designer know how much memory he needs to allocate for a given variable. For example, APPLESOFT requires 5 bytes of memory (or 5 memory location) to store a real variable such as 25.246 but only 2 bytes to store an integer variable i.e. a whole number with no fractional parts.

As a consequence, the general idea of learning programming, which is a fancy way of saying to tell the computer what you would like it to do, rapidly becomes an exercise in learning to wear a mental strait jacket. Due to the design of the programming language, one is forced to perform mental contortions to implement one's ideas in a program. There are many who will truly find such chores intrinsically interesting. These computer jocks will no doubt find fulfilling and successful careers as software engineers. For the average student, however, such exercises are downright boring.

Many high level languages are available on the Apple II and some are particularly suitable for special applications: FORTRAN for scientific computations, COBOL for business oriented programs and so on. Of these languages, LOGO occupies a very unique position, being a close cousin of LISP, a LISt Processing language used in Artificia Intelligence research.

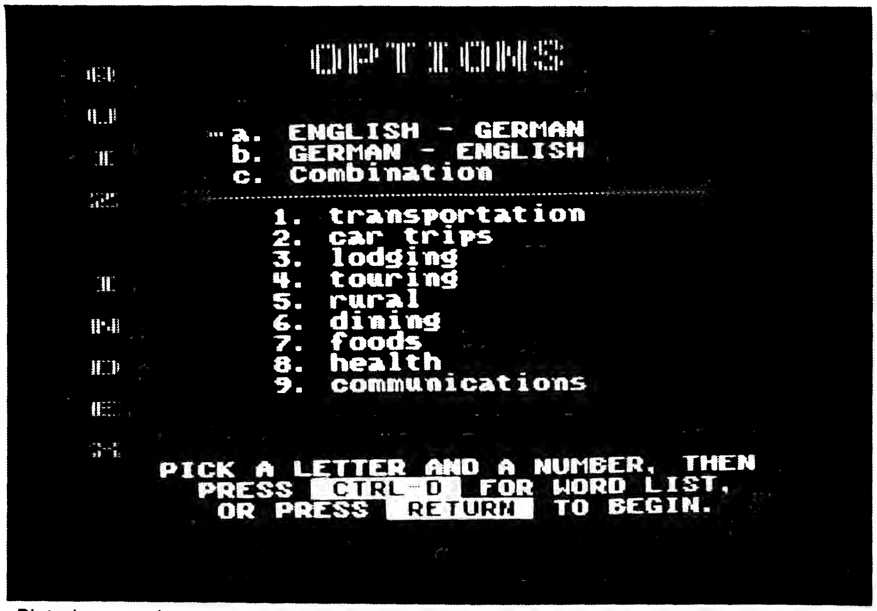
Like BASIC, LOGO is interpretive, that is, the compilation process mentioned earlier is not necessary. As a consequence the edit/run cycle is fast and easy, which is a very valuable asset in an educational environment. Furthermore, APPLE LOGO includes a superb full screen Editor which resides in memory (i.e. no need for time consuming disk accesses) and is a joy to use compared with the awkward Escape sequences of APPLESOFT BASIC. (An Editor is a program that allows one to make changes to the human understandable instructions to the computer. A full screen variety is one which permits one to move the cursor anywhere on the screen and to make insertions and deletions without tracing over with the right arrow and so on.)

LOGO is a modular or procedural language. For instance, if you want to draw a star on the screen you simply define a PROCEDURE and call it DRAWSTAR, say. Whenever you want to draw a star simply type DRAWSTAR in the immediate or deferred execution mode as in APPLESOFT BASIC. In a way, a PROCEDURE is like a subroutine with GOSUB or DEF FN in BASIC. However, a LOGO PROCEDURE may contain many lines of code, not just one line as in DEF FN. Also the variables within one PROCEDURE are totally separate from variables in other PROCEDUREs or in the calling program. That is, they can have different values even though they may have the same name. It is of course possible to have variables to take on the same value throughout a number of PROCEDUREs. PROCEDUREs can also take on any number variables as inputs e.g. to specify the colour and size to be used by DRAWSTAR.

Programming in LOGO becomes a matter of sorting your ideas into lots of simpler tasks or PROCEDUREs. These PROCEDUREs, once defined become, and can be used just like any RESERVED words like PRINT; they simply become a part of LOGO's vocabulary. In this manner, LOGO is open ended and extensible.

In fact, it is too simplistic to think of LOGO as just another programming language. When implemented, LOGO turns your computer into a learning machine that you, the programmer, can teach to perform specified tasks. More importantly, the interaction between man and machine is done in a manner that accomodates the human – interactive operation, excellent editing features, procedural structure, meaningful identifiers (not just the first two letters are significant) and so on.

In an educational setting, LOGO provides an excellent learning and exploring environment. For example, the teacher can define a set of suitable PROCEDUREs for the students. They can then be set the task of exploring the effects of and interaction between these PROCEDUREs. They can use these PROCEDUREs or construct new ones from them to solve problems. This way LOGO can be "taught" new tricks and the students learn the art of problem solving during this "teaching" process. The connection with Artificial Intelligence may now be a little clearer.



```
OPTIONS

a. ENGLISH - GERMAN
b. GERMAN - ENGLISH
c. Combination

   1. transportation
   2. car trips
   3. lodging
   4. touring
   5. rural
   6. dining
   7. foods
   8. health
   9. communications

PICK A LETTER AND A NUMBER, THEN
 PRESS  CTRL-D  FOR WORD LIST,
  OR PRESS  RETURN  TO BEGIN.
```

Plato in operation.

Another feature of LOGO that deserves mention is the way it treats data. LOGO does not make any distinction regarding variable types. A given variable can at one instance stand for a number but later on can denote a character string. For instance, a program may require a series of numerical data to be entered from the keyboard. In BASIC one would have to use special characters (Esc, null or control characters) to signal the end of data entry. In LOGO one can simply "no more" to terminate data entry. Of course one can achieve the same effect in BASIC with a few extra lines of code; but then we do not want to conform to the wishes of the machine, do we?

Apart from the usual arithmetic functions, LOGO has built in PROCEDUREs to manipulate data objects called WORDS and LISTS. A WORD is a string of characters. A LIST can be thought of as a collection of objects. These objects can be WORDS or even other LISTS. This feature makes LOGO very powerful in problems involving the manipulation of symbols. In contrast, BASIC only has STRING variables and built in functions only operate on one CHARACTER at a time. It is of course possible to simulate the WORD and LIST manipulating function in BASIC – but it then becomes a case of man being subservient to the machine again.

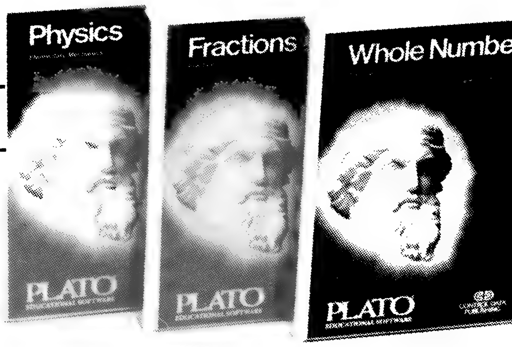
We hope to be able to discuss the LIST manipulating features of LOGO in a future article.

Turtle Geometry

One of the advantages of using LOGO as a learning environment is the availability of Turtle graphics.

Turtle graphics is simply a set of commands which moves a Turtle around the screen. The Turtle can be made to move forward, backward or turn by specified amounts. It can also be made to leave behind a trail of ink (!) of various colours to form a picture. Turtle graphics thereby solves the problem of What to Compute. It is very easy to make the Turtle do interesting things and, as we shall see, Turtle graphics itself is a very rich source of discoveries that extend far beyond child play.

You may recall the Cartesian



geometry you learned at school. It was named after the 15th century mathematician, Rene Descartes, who while watching the proverbial fly on the window pane, noted that it is possible to specify the position of the fly on the window by specifying the horizontal and vertical distances along the window frame relative to say the bottom left hand corner. Thus a point on the plane, (the position fly if you like) is determined by specifying a pair of numbers (x,y) called the coordinates of the point. For example, a point located at 3 units along the horizontal and 5 units along the vertical is given by (3,5). A straight line can be fixed by giving the two end points. A general curve, however, will be specified by mathematical relationship between the coordinates of points lying on the curve. For example, all points (x,y) lying on a circle of radius R centred at (a,b) satisfy the formula:

$$(x-a)^2 + (y-b)^2 = R^2$$

To specify a square of a given size with its sides parallel to the vertical and horizontal is very easy. However, to describe the same square which has been rotated through some given amount so its sides are no longer parallel to the vertical will require a knowledge of trigonometry.

The following LOGO PROCEDURE:

```
TO SQUARE :SIDE
PENDOWN
REPEAT 4 [FORWARD :SIDE
      RIGHT  9Ø]
END
```

will draw the generic square. This LOGO PROCEDURE is self explanatory to anyone who reads English. The Turtle is instructed to put the pen in the down position to draw and then move forward by an amount specified by the variable SIDE followed by a RIGHT turn of 9Ø degrees (a right angle). This action is to be repeated 4 times. To draw a square of 1ØØ units per side we simply say SQUARE 1ØØ to LOGO.

To draw a square that has been rotated by a given amount, we first set the initial position and heading of the Turtle by the appropriate amount and then ask LOGO to SQUARE 1ØØ.

Turtle geometry uses a LOCAL (Turtle's) view to specify a curve while Cartesian geometry uses a GLOBAL description of the curve. Technically, Turtle geometry uses differential equations or differential geometry while Cartesian geometry uses algebraic relations for the coordinates of the points on a curve.

Turtle geometry also introduces the idea of the topology of a curve. A square and a circle are topologically equivalent as one can be made from the other by simple stretching, and both are called TOPOLOGICALLY SIMPLE. On the other hand, a figure-of-eight is not topologically equivalent to a circle because the curve has crossed itself. These concepts lead to the TOTAL TURNING theorem for the Turtle. If the Turtle has traced out a topologically simple curve (no crossings) then the Turtle has turned through exactly 36Ø degrees. This is provided one assigns left hand turn to be positive and right hand turns to be negative or vice versa. If the Turtle had crossed its own path EXACTLY once then the Turtle has turned through Ø or 72Ø degrees. This depends whether the crossing is of the figure-of-eight or loop the loop type. The analagous theorem in Cartesian geometry is about the sum of interior angles of polygons – the interior angles of a triangle adds up to 18Ø degrees, a quadrangle to 36Ø degrees.

The book "Turtle Geometry: the Computer as a Medium for Exploring Mathematics" by H Abelson and A A diSessa makes compulsive reading for those who would like to delve into the riches of Turtle geometry. Do you want to know, for instance, how to use Turtle geometry to formulate Einstein's Theory of General Relativity?

Let me leave you with the following to explore:

```
TO FIGURE :STEP :SINC :ANGLE
:AINC
PENDOWN
FORWARD :STEP
RIGHT :ANGLE
FIGURE   :STEP+:SINC   :SINC
ANGLE+:AINC :AINC
END
```

Play with this PROCEDURE using the invocation:

```
FIGURE 1Ø 3 12Ø 3Ø
```

Try first with the increments SINC AND AINC set to zero. Try positive and negative increments. Set one of the increments to zero. And observe carefully the relation between the inputs and the figures generated, and note the effects of inputs that will divide exactly into 9Ø, 18Ø, 27Ø, 36Ø. Above all, experiment.

Happy bit grinding! □

MICRO NATIONWIDE

Manufacturers and suppliers of quality software wish to announce the arrival of:–

SOUTH DAKOTA

AN ECONOMIC AND MATHEMATICS SIMULATION.

For this and other quality educational programs check with your local distributor or:–

MICRO NATIONWIDE
P.O. BOX 653,
MONA VALE, 2103.

A SYMBOL OF QUALITY EDUCATIONAL SOFTWARE

Disk to tape dump utility

by Peter Mearing

Backing up disks against inevitable data-destroying disasters is a necessary pasttime of all people involved in the running of a computer. This applies to almost any size computer, using floppy or hard disks.

I have had cause to regret not taking the precaution of having kept a copy of valuable programs in a safe environment. Fortunately, I have, as yet, not had any severe losses of data, but sooner or later that too is bound to happen.

When I first started programming and using the Apple, my storage of disks was quite small and to have backups of each one in current use was not too expensive. However, as the number of disks in active use started to climb to above the 15Ø mark, the sheer cost of keeping a backup of each was becoming prohibitive, even at the cost price of a disk!

I decided, therefore, to look into alternative ways of disk backup. The cheapest, most effective way I found was to use cassette tape of good quality and a domestic tape recorder. A good C-9Ø tape doesn't cost much and, being able to fit six disks onto a tape, this represents a minimal cost per backed up diskette – a quarter of the cost of using disks, or less if normal retail disks are used. If stereo channels on the tape recorder are used separately, then disks will fit onto a tape. I have always used good quality C9Ø tapes because of their reliability, something that cheap tapes definitely lack!

All good things have a price, though, and this facility is no exception! The main price is time. It takes approximately 13 to 15 minutes to back up each disk and the same time to retrieve it. Admittedly this could be shortened considerably, and the amount of storage also improved by using some method other than the Monitor SAVE and READ routines this program uses. However, that would involve a much greater cost because a separate interface must be bought or built. The only bit of work involved in this system is a small optional circuit described below that can be built to control the tape recorder.

The total cost involved in this backup facility is very little for the circuit components and plugs (excluding, of course, the tape recorder). All things considered, the backup system described here is best suited to long term semi-permanent storage of information such as program disks and old data disks whose information should be preserved. The system can back up DOS, CP/M and Pascal disks with equal ease because it operates as a data mover rather than a file mover. Protected disks are not amenable to being copied by this program.

Program listing

```
TRACKS R/W      7 Track Read/Write Program.       03-Nov-1983 #0007 PAGE 1


----- NEXT OBJECT FILE NAME IS TRACKS R/W.OBJ0
1400:                     ORG  $1400

            ******************************************************
1400:       * TRACKS R/W: A Program to Read/Write 7 Disk Tracks.  *
            ******************************************************

1400:       *** ZERO PAGE USAGE

0003:       PTR      EQU  $3          Work Pointer
0002:       OPER     EQU  $2          Operation to be performed
0005:       ERR      EQU  $5
0001:       SECTOR   EQU  $1
0000:       TRACK    EQU  $0
0048:       PREG     EQU  $48         MONITER P Register Save Area
0087:       BELL     EQU  $87

1400:       *** OTHER ADDRESSES

0302:       BUFFER   EQU  $302        Sector data buffer
0300:       T        EQU  $300
0301:       S        EQU  $301        Track and Sector counters
FDED:       COUT     EQU  $FDED
03E3:       LOCIOB   EQU  $3E3        Locate RWTS Parmlist Subroutine
03D9:       RWTS     EQU  $3D9        RWTS Routine

1400:       *** RWTS IOB DEFINITION

0000:                DSECT
0000:       IOBIOB   DS   1           IOB Type ($01)
0001:       IOBSLT   DS   1           SLOT*16
0002:       IOBDRV   DS   1           DRIVE
0003:       IOBVOL   DS   1           VOLUME
0004:       IOBTRK   DS   1           TRACK
0005:       IOBSEC   DS   1           SECTOR
0006:       IOBDCT   DS   2           Address of DCT
0008:       IOBBUF   DS   2           Address of Buffer
000A:       IOBSIZ   DS   2           Sector Size
000C:       IOBCMD   DS   1           Command Mode

000D:       *** COMMAND MODE CODES

000D:       * $00 EQU  NULL Command
000D:       * $01 EQU  READ Command
000D:       * $02 EQU  WRITE Command
000D:       * $04 EQU  FORMAT Command
000D:       IOBRCD   DS   1           Return Code
```

listing continues

System Overview

The basic backup system consists of four items: the Apple computer with a disk drive, a tape recorder plugged into the Tape I/O of the computer, the system program and a small, three-component circuit that plugs into both the game control socket and the tape recorder to control the tape recorder run/stop function. The circuit is not essential but it is very useful to keep recordings compact.

The Apple II Monitor SAVE and LOAD routines are used to write to and read from the tape and a machine language program called TRACKS R/W (a modified and extended version of the program T/S MOD) is used to read and write the disk.

The circuit diagram of the tape recorder on/off controller interface is given in Fig 1. It takes its input from the game controller annunciator output ANØ and powers a relay to switch the tape recorder on or off. Most tape recorders have some form of remote on/off switching mechanism. If yours does not, it doesn't matter as the program will work just as well without it. The only drawback is that more tape will be used as the recorder will be left running while disk reads and writes take place. Since the disk reads and writes are of short duration, the extra tape playing time is easily accommodated on the tape.

The circuit is best constructed on a dual-in-line (DIL) plug that fits directly into the games socket, or on a small piece of vero board, or a tag strip with wires going to plugs in the games socket and the recorder. The biggest item in the circuit is the relay, but I used a subminiature encapsulated relay with mercury-wetted contacts that was about two-thirds the size of one of the memory integrated circuits in the Apple. This works on 5 volts and so the 5V supply on the games output socket was used to power it. In brief, Pin 1 carries +5V, Pin 8 carries the ground rail and Pin 15 the ANØ output.

The transistor can be any NPU switching transistor of reasonable gain; I used a BC169C, and the resistor can be Ø.25 or Ø.125 watt 5% 1ØØØ ohm resistor. The relay I used is fairly expensive but a larger one can be obtained for much less.

listing continued

```
000E:         *** RETURN CODES

000E:         * $00 EQU  All well
000E:         * $10 EQU  Write Protected
000E:         * $20 EQU  Volume Mismatch
000E:         * $40 EQU  Drive Error
000E:         * $80 EQU  Read Error
000E:         IOBTVL  DS   1          True Volume
000F:         IOBPSL  DS   1          Previous slot
0010:         IOBPDR  DS   1          Previous Drive
1400:                 DEND

1400:         *** SAVE ALL REGISTERS

1400:48               PHA              This is where it all starts
1401:98               TYA
1402:48               PHA
1403:8A               TXA
1404:48               PHA

1405:A9 0F            LDA  #$0F       Start with Sector F
1407:85 01            STA  SECTOR     Pointer.
1409:A9 20            LDA  #$20       Reset Buffer.
140B:8D 02 03         STA  BUFFER
140E:A9 07            LDA  #$07
1410:8D 00 03         STA  T          Do 7 Tracks
1413:A9 10            LDA  #$10
1415:8D 01 03         STA  S          And 16 Sectors.

1418:         *** FILL IN RWTS IOB

1418:20 E3 03         JSR  LOCIOB     Locate RWTS Parmlist
141B:84 03            STY  PTR        and save pointer
141D:85 04            STA  PTR+1

141F:A5 00    START   LDA  TRACK      get TRACK to READ/WRITE
1421:A0 04            LDY  #IOBTRK    and store in RWTS list
1423:91 03            STA  (PTR),Y
1425:A5 01            LDA  SECTOR     Get Sector to Read
1427:A0 05            LDY  #IOBSEC
1429:91 03            STA  (PTR),Y    Store in RWTS List
142B:A0 08            LDY  #IOBBUF
142D:A9 00            LDA  #$00       Store Buffer pointer in List
142F:91 03            STA  (PTR),Y
1431:C8               INY
1432:AD 02 03         LDA  BUFFER
1435:91 03            STA  (PTR),Y

1437:A5 02            LDA  OPER       Get Command Mode
1439:A0 0C            LDY  #IOBCMD    and store in List
143B:91 03            STA  (PTR),Y

143D:A9 00            LDA  #0         This will allow Any Volume
143F:A0 03            LDY  #IOBVOL
1441:91 03            STA  (PTR),Y    Store in List

1443:         *** NOW CALL RWTS TO READ/WRITE THE SECTOR

1443:20 E3 03         JSR  LOCIOB     Reload pointers
1446:20 D9 03         JSR  RWTS       Off we go!
1449:A9 00            LDA  #0         If we exit now, we must
144B:85 48            STA  PREG       Fix P reg so DOS is Happy
144D:85 05            STA  ERR        Store OK code for Calling Program
144F:90 0E            BCC  NEXT       All is Well, do next sector
1451:         *** ERROR OCCURED!

1451:A9 87            LDA  #BELL      BEEP the speaker
1453:20 ED FD         JSR  COUT

1456:A0 0D            LDY  #IOBRCD
1458:B1 03            LDA  (PTR),Y    Get RWTS Return Code
145A:85 05            STA  ERR        Store Error Code for Call Program
145C:4C 7C 14         JMP  EXIT

145F:EE 02 03 NEXT    INC  BUFFER     Step to next Buffer section.
1462:C6 01            DEC  SECTOR     Count Down Sectors.
1464:10 1C            BPL  CONT       Next Track.
1466:E6 00            INC  TRACK      Count Down Tracks.
```

listing continues

Driving Programs

The system program consists of two parts; a BASIC driving program (see Fig.2: Program DUMP) and a machine-code disk-access subroutine (see Fig. 3: Program TRACKS R/W). I will first describe the BASIC driving program and then fill in details on the machine code routine used.

The program uses one memory buffer, viz a tape buffer 28672 ($7ØØØ) bytes long able to contain up to seven disk tracks at a time.

When a disk is dumped to tape, each sector is read into the appropriate part of the tape buffer until seven tracks have been read. The tape buffer is then written to the tape. The process is repeated until the entire disk has been dumped.

The reverse happens when retrieving disk data from the tape. Seven tracks are read in from the tape and then sector sized segments are written to disk until all seven tracks are written; this procedure being repeated until the entire disk has been retrieved.

Seven tracks per transfer was chosen because the memory required to house all this data ($7ØØØ or 28k bytes) is near the capacity of the normal 48k Apple and, seven can be divided nicely into 35 giving five full transfers. Whether a disk contains 35 or 36 tracks (as may be the case in disks modified using the SPACE program), 35 tracks are transferred; track Ø is ignored in the case of 36 track disks as, in any case, track Ø can contain only part of the DOS BOOT image and will therefore be present on any initialised disk used when retrieving the disk from tape.

Please note that only formatted disks can be used for the retrieval to be successful; virgin or degaused disks will cause write errors and abort the program operation. With the retrieval of Pascal and CP/M disks, please be sure to use disks formatted under the appropriate system. These disks will contain 35 tracks in total and it is essential to use the 35 track mode (NOT 36 track mode) as track Ø will contain valuable information on these disks.

The machine code program used to read and write to the disk is the most efficient I have managed to devise so far, and uses optimum skewing of sectors and eliminates the use of intermediate buffers which are the basis of most of the DOS speed-up programs available.

The tape read/write times cannot really be reduced unless, as mentioned before, a completely different hardware system is used.

Needless to say, this program can be improved. Disk reads could take place to see what kind of disk is in the drive and parameters set up accordingly and stored on tape to be read on retrieval so all the parameters are correct for the specific disk record. An automatic disk formatting program could be added if the RWTS comes up against write errors.

listing continued

```
1468:A9 0F              LDA  #$0F      Else Start all Over.
146A:85 01              STA  SECTOR    Reset Sector count.

146C:A5 02              LDA  OPER      Decide on appropriate Display.
146E:C9 02              CMP  #$02      Is it a WRITE?
1470:F0 13              BEQ  WRITE     Yes, Branch
1472:A9 D2              LDA  #'R'      No! We are READING!
1474:20 ED FD    DECT   JSR  COUT

1477:CE 00 03           DEC  T         Next Track
147A:D0 06              BNE  CONT      Continue if not yet done.

147C:            *** EXIT TO CALLER AFTER RESTORING ALL REGISTERS

147C:68          EXIT   PLA             Restore Registers.
147D:AA                 TAX
147E:68                 PLA
147F:A8                 TAY
1480:68                 PLA
1481:60                 RTS

1482:4C 1F 14    CONT   JMP  START

1485:18          WRITE  CLC             Prepare to Branch after we
1486:A9 D7              LDA  #'W'      Load a 'WRITE' Symbol
1488:90 EA              BCC  DECT      Branch back

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

Symbol Table — sorted by symbol

```
   87 BELL          0302 BUFFER         1482 CONT          FDED COUT
 1474 DECT            05 ERR            147C EXIT            08 IOBBUF
   0C IOBCMD      ?   06 IOBDCT       ?   02 IOBDRV      ?   00 IOBIOB
?  10 IOBPDR      ?   0F IOBPSL           0D IOBRCD          05 IOBSEC
?  0A IOBSIZ      ?   01 IOBSLT           04 IOBTRK      ?   0E IOBTVL
   03 IOBVOL        03E3 LOCIOB         145F NEXT            02 OPER
   48 PREG            03 PTR            03D9 RWTS            01 SECTOR
 0301 S             141F START          0300 T               00 TRACK
 1485 WRITE
```

Symbol Table — sorted by address

```
   00 TRACK       ?   00 IOBIOB       ?   01 IOBSLT          01 SECTOR
   02 OPER        ?   02 IOBDRV           03 PTR             03 IOBVOL
   04 IOBTRK          05 ERR              05 IOBSEC      ?   06 IOBDCT
   08 IOBBUF      ?   0A IOBSIZ           0C IOBCMD          0D IOBRCD
?  0E IOBTVL      ?   0F IOBPSL       ?   10 IOBPDR          48 PREG
   87 BELL          0300 T              0301 S             0302 BUFFER
 03D9 RWTS          03E3 LOCIOB         141F START         145F NEXT
 1474 DECT          147C EXIT           1482 CONT          1485 WRITE
 FDED COUT
```

However, increased sophistication does imply increased complexity.

An interesting phenomenon occurs should you press RESET during a tape READ or WRITE. Subsequent output to the screen may turn out to be garbage! To return things to normal simply type NORMAL (less than sign) cr (greater than sign) while in APPLESOFT command mode.

The TRACKS R/W Program works as follows:

Before CALLing the program, the desired starting track is stored in memory location Ø. The desired function to be performed, 1 for a read and 2 for a write is stored in memory location 2.

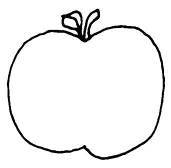
On CALLing the program, all registers are first saved after which a jump is performed to LOCIOB, a DOS routine which locates the position of the RWTS (Read Write a Track Sector – a DOS routine) Parameter list. On return the parameter list is filled in as regards desired track, sector and buffer address and RWTS is called to read or write the sector in question. On exiting RWTS a check is made to see whether an ERROR was encountered and if so, an error code is stored in memory location 2 for the calling program to do something about and execution aborted.

With each read or write, the status display is updated with the appropriate symbol.

Conclusions

The backup system has worked well on my system for more than a year with minimal problems and many revisions to improve efficiency. The only real drawbacks have been speed (lack of!) and the inability to verify a copy of a disk (though I have not had a dud tape copy yet). All my disks have worked well on retrieval from tape.

The main advantages of operating this system are its cost effectiveness and simplicity. As entire disks are backed up, it is desirable to use full disks, otherwise time is wasted backing up nothing. I hope you get as much use out of it as I have!



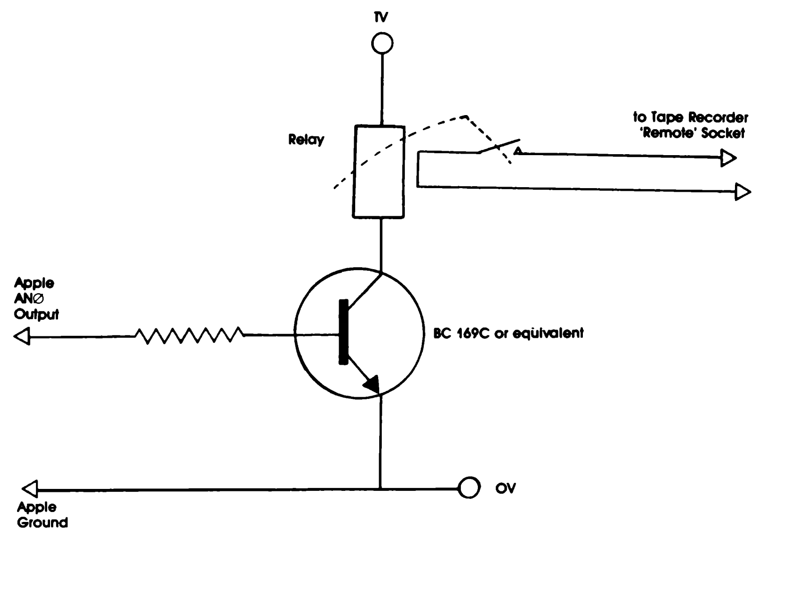


Fig 1 Recorder Control Circuit
IV depends on the relay used — it can be the Apple 5V games supply if a 5V relay is used.

The DUMP program works as follows:

Lines 2Ø to 3Ø: Initialise various constants and CALL locations and Load in the TRACKS R/W routine. The number of tracks per disk is set to 35, ie the last track number is 34. Because I use 36-track disks, a track number select option is included in the memory should you want to use 36-track disks. If you want 36-tracks by default then line 2Ø must be altered so that LT%=35 and TR%=1.

Lines 5Ø to 98: Print menu on the screen and allow keyboard choice of desired function. To quit the program, an extra menu choice could be put in or RESET must be pressed. The total number of tracks can be flipped between 35 and 36 depending on the known number on the disk. The tape recorder can also be toggled on or off to allow rewinding, cueing, etc.

Lines 1ØØ to 21Ø: The Tape to Disk Retrieval routine. Tracks are read from the tape in lots of seven by CALLING the MONITOR tape READ routine after setting up the appropriate page 1 conditions and then written a blank initialised disk sector by sector. A status display is maintained to inform the user of progress. If tape read errors then ERR message will appear in the status display and the retrieval process should be stopped and retried. The tape recorder is switched from run to stop and vice versa by accessing the appropriate memory locations controlling the ANØ Output on the games I/O socket if the tape control interface is installed. A short delay is instituted in line 125 to allow the recorder to come up to speed before reading is commenced.

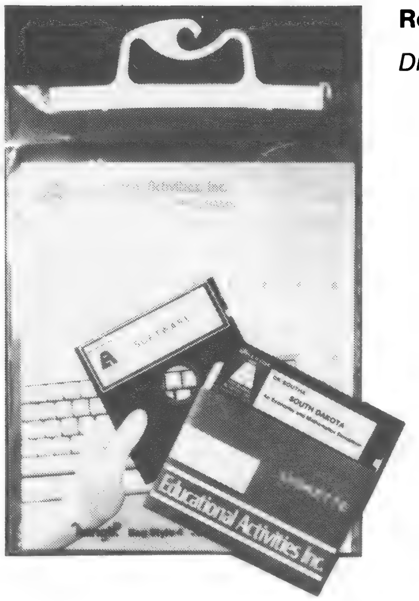
Lines 3ØØ to 41Ø: The Disk-to-Tape Dump routine. Tracks are read in lots of seven, sector by sector, into the tape buffer memory area and written to the tape en masse using the MONITOR tape SAVE routine. Both the SAVE and READ routines require the start and end addresses POKEd into memory at locations $3C to $3F. The MOVE routine requires the origin start and end addresses at locations $3C to $3F, and the destination start address in locations $4Ø and 41.

Lines 45Ø to 49Ø: Set up the status display. It is individualised for either a 35 or 36 track retrieval/dump.

Line 5ØØ: Gives a pause with the tape running to provide a gap on the tape between disk records.

Line 7ØØ: A simple error handling routine to cope with errors encountered by TRACKS R/W. □

Farming in "South Dakota"



Reviewed by Gary Williams

Distributed by Micro Nationwide

South Dakota is an economic and mathematics simulation game developed by Educational Activities Inc and distributed by Micro Nationwide. To use this program you will need an Apple II, one disk drive and 48K of RAM.

My first impressions of the program were of the packaging, which is nothing to write home about. Have you ever seen those cheap shirts that come in plastic bags with a coat hanger type seal at the top? Great if you have a nice container to store your disks in, but not very thoughtful for those who rely on receiving their disks in a nice hard box, which keeps them safe when not in use. The other complaint is in the documentation. Not that it isn't well written, it is. It's just that it is supplied loose with no binder, although the makers have thoughtfully punched holes so it can be inserted in the binder of your choice.

Included in the documentation are various tables you can photostat and use to keep a record of the statistics of the game as you progress. This is essential if you are to use the program as it was intended – as an educational tool.

The object of the game is to run a farm in South Dakota for a period of ten years (game turns). Sounds easy? Think again. On my first attempt I was bankrupt within three years. It definitely pays to utilise the tables supplied to keep track of money in and out, etc. Although I found it more fun to simply waffle through using intuition.

'Keep a record of the statistics of the game as you progress'

When you finally make it through the ten years you are given your initial assets as compared to your final assets. Unfortunately, you will be in for a rude shock if you haven't kept a record of everything as suggested in the manual. The first time I managed to complete the ten year term I ended up with half the assets I started with, which is most disconcerting after spending all those years working my fingers to the keyboard.

Each year (game turn) you must decide which grain to sow from a choice of five (barley, wheat, oats, rye and corn). Then you must purchase enough seed to sow your acreage. You initially have 15Ø acres and $5Ø Ø to build on. Depending on how many acres you wish to sow, you may need to hire extra help (the cost of which increases each year). As well, a large amount of cash is subtracted from your kitty to feed and clothe you and your family over the growing and winter seasons.

Unfortunately you will find that your $5ØØ capital investment is nowhere near enough to cover the cost of your first year as a farmer. And so enters your smiling bank manager. You can borrow as much as you like up to $9,9ØØ but, as with all banks, you will have to pay interest, at 12% per annum.

If, like me, you have a natural aversion for banks, you can always sell some of your land. This can be a good thing as the market price of land varies between $199 and $699. So if your green finger is looking a bit brown you may be able to make up the deficit in real estate speculation.

A nice touch is the musical interlude which plays during the growing period. This is the time you start pruning your finger nails down to the cuticles hoping you have made the right decision as to the crop chosen.

After the growing period is over you are told the new selling price of the five grains, which are affected by a variety of things. These range from floods and droughts to pesticide scares and the economic climate in the USSR. Your grains are automatically sold and the profits, less tax, are added to your kitty. Then you must pay back half of your loan, plus interest, to the bank. Like all banks they can get very stroppy if you haven't the funds to pay them back.

Another nice touch is the fact that you can purchase things like chickens, cows, tractors and combine harvesters (if you can scrape up the $2Ø,ØØØ) to help ease your burden. The chickens and cows are relatively inexpensive and save you a bit when the time comes to buy food for your family. The tractor and combine harvester seem very expensive at $13,ØØØ and $2Ø,ØØØ, but they take the place of numerous farm hands you would normally have to hire twice each year, when sowing and harvesting. Also, their value is included in your assets which unfortunately depreciate in value each year. If you have taken a loan to buy your farm machinery (which I'm certain you would have done) you may find yourself in some strife if your crops fail. If this is the case, the bank may very well take away your tractor and/or combine to meet your commitment.

The graphics left me a bit cold, and are restricted to a static picture of your farmhouse in three different situations. The first is of the farmhouse in summer, the second is the same but in winter and the third

'On my first attempt I was bankrupt within three years'

is one you will not want to see (a derelict farm house with a foreclosure sign on it). As well, there is graphic representation of the five grains on the left of the screen when in the playing mode.

As an educational tool South Dakota would be invaluable to those people learning farm management, but as a pure game it can be very frustrating. A feature of the program that makes it appealing as an educational tool and annoying for the game playing purists is the "press space bar to continue" feature. This is to allow time to keep your records up to date after each transaction. I would like to see the program incorporating a choice of whether to include this feature or not.

South Dakota costs $59.ØØ and I'm told there will be an Australian version for sale in the not too distant future. The ability to save a game at any stage would be an advantage, and I'm told the distributors intend to do something about that, too. This means that South Dakota, as a farm management game, could be used as an "if-then" type set-up. Allowing the users to say for example "OK, here's where we stand now, how about we try this and see what happens" and if that proved disastrous they could reload where they started from and give something else a try.

So there you have it, South Dakota is a brilliant educational tool that doubles as a down to earth game (excuse the pun). It can be played with little knowledge of farming and still thoroughly enjoyed. □

Ease of use	****
Graphics	**
Sound	**
Value for money:	
as a game only	***
educational	*****
Sustained interest	***

Note: the above ratings are out of a possible five stars.

More disk storage under DOS 3.3

by Peter Meiring

When I first bought an Apple II about two years ago, I had visions of storing vast amounts of information on disk with near instant recall, correlation and computation of all of it. Alas, I found that Standard DOS 3.3 is restricted to only 496 sectors of disk storage space (124k), less any housekeeping sectors that are necessary with all files stored. Undaunted, I decided to investigate methods of increasing storage space on disks and came up with three basic methods, namely:

1. Pre-storage packing of information and more advanced database management.
2. Deleting the diskette DOS (Disk Operating System) image and using the space it occupied or using sectors unused by the DOS if a DOS image is required to let the disk be a BOOT disk. The same applies to the space occupied by the catalog on track 17 ($11) if one only intends to use a strictly limited number of files on the disk.
3. Using Track 35 (normally not initialised by INIT).

In this article I shall discuss the latter two methods (pre-storage packing and advanced database management are enough for another complete article!). First I will discuss some of the relevant details pertaining to the Apple DOS and then give some sample programs to illustrate implementation of the ideas. Using both methods, up to 544 (136k) sectors are usable as opposed to 496 (124k) on a normal DOS disk. If a disk is used for a maximum of 7 data files, a further 14 sectors can be used by freeing them in Track 17, giving a total of 599 (139.5k) sectors usable, 62 (15.5k) sectors more than normal. This corresponds to a 12.5% increase in space.

Most numbers and values are given in decimal and hexadecimal; the latter numbers are preceded by the symbol $.

Using DOS Sectors for Storage

The DOS organises its disks by formatting them to have 35 tracks (Ø-34 or $Ø-$22) of 16 sectors each. Of these, Tracks Ø, 1 and 2 are used by the DOS to store an image of itself to copy into the Apple's memory during the BOOT process and Track 17 ($11) is used to store the CATALOG of the disk as well as its Volume Table of Contents (VTOC).

When the DOS is storing information on a disk and has filled up the sector it is currently writing to, it jumps to a routine that reads the VTOC. It examines this sector and finds which sectors are still unoccupied so it may set its pointers to write the next lot of information to the next unoccupied sector. This sector is then marked in the VTOC as used and is then protected from being overwritten by subsequent information.

The sections of the VTOC responsible for this housekeeping are the track bit maps located from byte number $38 to $FF. Each of these bit maps are four bytes long, giving, in theory, enough space in the VTOC for 5Ø tracks but only 35 are used normally. Of the four bytes, only two (16 bits) are necessary to map any given track as each track contains 16 sectors. Each of the bits in the two used bytes corresponds to one sector in the track; if a bit is set to a '1' then that sector is not currently used by any stored program; otherwise, if set to 'Ø', it is in use.

The only protection for the sectors used by any program, or the DOS for that matter, therefore lies in the exact status of the VTOC situated on Track 17, Sector Ø. If the VTOC is damaged in any way, no program on the disk can be considered as safe unless the disk is write protected or the VTOC is restored to its former glory! Similarly, if the bit maps of Tracks 1 & 2 are set to 11111111 11111111 ØØ

ØØØØØØ ØØØØØØØØ ($FF $FF $ØØ $ØØ), then the DOS will consider them vacant and use them when needed. The only drawback is a disk so modified will not be suitable for BOOTing the Apple, but this is no real price to pay for the extra sectors of storage space (up to 12.5k or 12,8ØØ bytes) since data disks do not, in general, need a copy of the DOS on board.

Track Ø, Sector Ø cannot, unfortunately, be used in the same way because of the way the DOS works.

What happens can be explained as follows:

The basic structure of a file on disk consists of three parts:

1. Catalog entry consisting of a Track/Sector list Pointer, File type code, File name and a sector count.
2. Track/Sector list, being a list of track/sector pairs in which the file data resides on disk.
3. Data pointed to by 2. above.

When the DOS is asked to retrieve a file, it first looks in the catalog (Track 17, Sectors 1 to 15 – starting in Sector 15) to find the file in question and retrieves the contents of the Catalog entry. Using the Track/Sector List Pointer, it then accesses the Track/Sector list and retrieves the first Data Track/Sector pointer which it uses to retrieve data. Upon complete retrieval of that sector, it jumps back to the Track/Sector list, gets the next pointer, retrieves the data and so on until it retrieves a track/sector pair both equal to zero, signalling the end of the Track/Sector list.

It can be seen that, should one try to use track Ø, sector Ø in the track/sector list, the DOS will promptly abort retrieval of information. This is due to DOS being under the false impression that the last sector of information had already been retrieved and the file was at an end! Hence, no user access to track Ø, sector Ø! Hence, theoretically, 15 sectors are available on track Ø. In fact, the Apple DOS is a little lazy and does not bother checking **both** track **and** sector pointers; making do with just the Track pointer. If it encounters a zero track pointer then it assumes it has reached the end of the file! Hence track Ø is best left alone.

In summary, all one has to do to

Program listing 1

```
FORMAT          Track Formatting Program            05-05-1983 #00018 PAGE 1

----- NEXT OBJECT FILE NAME IS FORMAT.OBJ0
1300:                    ORG  $1300

                   *****************************************************
1300:              *  FORMAT : A Program to INIT any Track on a diskette *
                   *****************************************************

1300:              *** ZERO PAGE USAGE

0003:              PTR      EQU  $3
0000:              TRACK    EQU  $0         Desired Track.
0001:              VOLUME   EQU  $1
0002:              OPER     EQU  $2         Desired Operation.
002D:              SECFND   EQU  $2D
003E:              AA       EQU  $3E
0041:              VOL      EQU  $41
0044:              TRK      EQU  $44
0045:              SYNCNT   EQU  $45        Sync count used by DSKF2
0048:              PREG     EQU  $48        MONITER P Register Save Area
0087:              BELL     EQU  $87        ASCII Bell

1300:              *** OTHER ADDRESSES

03E3:              LOCIOB   EQU  $3E3       Locate RWTS Parmlist Subroutine
03D9:              RWTS     EQU  $3D9       RWTS Routine
0578:              RTRYCNT  EQU  $578       Retry count for DSKF2
BB00:              NBUF1    EQU  $BB00      Primary Sector Buffer
BC00:              NBUF2    EQU  $BC00      Secondary Sector Buffer
B8DC:              READ16   EQU  $B8DC      Read Data Field Routine
B944:              RDADR16  EQU  $B944      Read Address Field Routine
BF0D:              DSKF2    EQU  $BF0D      Format One Track Routine
FDED:              COUT     EQU  $FDED      COUT

1300:              *** RWTS IOB DEFINITION

0000:                       DSECT
0000:              IOBIOB   DS   1          IOB Type ($01)
0001:              IOBSLT   DS   1          SLOT*16
0002:              IOBDRV   DS   1          DRIVE
0003:              IOBVOL   DS   1          VOLUME
0004:              IOBTRK   DS   1          TRACK
0005:              IOBSEC   DS   1          SECTOR
0006:              IOBDCT   DS   2          Address of DCT
0008:              IOBBUF   DS   2          Address of Buffer
000A:                       DS   2
000C:              IOBCMD   DS   1          Command Mode
000D:              IOBRCD   DS   1          Return Code
000E:              IOBTVL   DS   1          True Volume
000F:              IOBPSL   DS   1          Previous slot
0010:              IOBPDR   DS   1          Previous Drive
1300:                       DEND


1300:              *** SAVE ALL REGISTERS

1300:48                     PHA
1301:98                     TYA
1302:48                     PHA
1303:8A                     TXA
1304:48                     PHA

1305:              *** USE RWTS TO POSITION THE ARM TO DESIRED TRACK

1305:20 E3 03               JSR  LOCIOB     Locate RWTS Parmlist
1308:84 03                  STY  PTR        and save pointer
130A:85 04                  STA  PTR+1
130C:A5 00                  LDA  TRACK      get TRACK to READ/WRITE
130E:A0 04                  LDY  #IOBTRK    and store in RWTS list
1310:91 03                  STA  (PTR),Y
1312:A9 00                  LDA  #$00       NULL Operation
1314:A0 0C                  LDY  #IOBCMD    and store in list
1316:91 03                  STA  (PTR),Y
1318:A9 00                  LDA  #0         any VOLUME will do
131A:A0 03                  LDY  #IOBVOL
131C:91 03                  STA  (PTR),Y
131E:20 E3 03               JSR  LOCIOB     Reload pointer to parms
1321:20 D9 03               JSR  RWTS       Call RWTS
1324:BD 89 C0               LDA  $C089,X    Leave Drive ON
```

```
1327:                *** ESTABLISH ENVIRONMENT FOR DISKF2 ROUTINE

1327:A5 00                    LDA  TRACK
1329:85 44                    STA  TRK         Pass Track to DISKF2
132B:A5 01                    LDA  VOLUME      and VOLUME
132D:85 41                    STA  VOL
132F:A9 AA                    LDA  #$AA        Store constant for ZERO PAGE TIMING
1331:85 3E                    STA  AA
1333:A9 28                    LDA  #$28        Start with 40 syncs
1335:85 45                    STA  SYNCNT
1337:A0 56                    LDY  #$56
1339:A9 00                    LDA  #$00
133B:99 FF BB        ZNBUF2   STA  NBUF2-1,Y ZERO Secondary Buffer
133E:88                       DEY
133F:D0 FA                    BNE  ZNBUF2
1341:99 00 BB        ZNBUF1   STA  NBUF1,Y     and Primary Buffer
1344:88                       DEY
1345:D0 FA                    BNE  ZNBUF1
1347:20 0D BF                 JSR  DSKF2       FORMAT Track and VERIFY
134A:A9 08                    LDA  #$08        in case of ERROR
134C:B0 19                    BCS  ENDERR      ERROR

134E:                *** READ SECTOR 0 OF NEW TRACK TO VERIFY FORMATTING

134E:A9 30                    LDA  #$30        No, Double Check Track
1350:8D 78 05                 STA  RTRYCNT     allowing 48 retries
1353:38              NOGOOD   SEC
1354:CE 78 05                 DEC  RTRYCNT     Count the Retries!
1357:F0 0E                    BEQ  ENDERR
1359:20 44 B9                 JSR  RDADR16     READ an Address Field
135C:B0 F5                    BCS  NOGOOD
135E:A5 2D                    LDA  SECFND      Is this Sector ZERO?
1360:D0 F1                    BNE  NOGOOD      No, try again
1362:20 DC B8                 JSR  READ16      Yes, READ Data Field
1365:90 0F                    BCC  DONETRK     All is well
1367:A0 0D           ENDERR   LDY  #IOBRCD     Create Return Code environment
1369:91 03                    STA  (PTR),Y

136B:                *** ERROR OCCURED!!

136B:A9 87                    LDA  #BELL       BEEP the speaker
136D:20 ED FD                 JSR  COUT
1370:A0 0D                    LDY  #IOBRCD
1372:B1 03                    LDA  (PTR),Y     Get RWTS Return Code
1374:85 02                    STA  OPER        Error Code for call Program

1376:                *   EXIT TO CALLER

1376:BD 88 C0        DONETRK  LDA  $C088,X     Turn OFF the Drive
1379:A9 00                    LDA  #$00
137B:85 48                    STA  PREG        Clear P register for DOS

137D:                *** Restore all Registers

137D:68                       PLA
137E:AA                       TAX
137F:68                       PLA
1380:A8                       TAY
1381:68                       PLA
1382:60                       RTS

*** SUCCESSFUL ASSEMBLY: NO ERRORS

SYMBOL TABLE    SORTED BY SYMBOL                      05-05-1983 #00018 PAGE 4

   3E AA               87 BELL          FDED COUT              1376 DONETRK
 BF0D DSKF2          1367 ENDERR       ?  08 IOBBUF              0C IOBCMD
?  06 IOBDCT        ?  02 IOBDRV       ?  00 IOBIOB           ?  10 IOBPDR
?  0F IOBPSL           0D IOBRCD       ?  05 IOBSEC           ?  01 IOBSLT
   04 IOBTRK        ?  0E IOBTVL          03 IOBVOL            03E3 LOCIOB
 BB00 NBUF1          BC00 NBUF2         1353 NOGOOD              02 OPER
   48 PREG             03 PTR           B944 RDADR16           B8DC READ16
 0578 RTRYCNT        03D9 RWTS            2D SECFND              45 SYNCNT
   00 TRACK            44 TRK             01 VOLUME              41 VOL
 1341 ZNBUF1         133B ZNBUF2       SYMBOL TABLE      SORTED BY ADDRESS

            05-05-1983 #00018 PAGE 5

?  00 IOBIOB           00 TRACK        ?  01 IOBSLT              01 VOLUME
?  02 IOBDRV           02 OPER            03 PTR                 03 IOBVOL
   04 IOBTRK        ?  05 IOBSEC       ?  06 IOBDCT           ?  08 IOBBUF
   0C IOBCMD           0D IOBRCD       ?  0E IOBTVL           ?  0F IOBPSL
?  10 IOBPDR           2D SECFND          3E AA                  41 VOL
   44 TRK              45 SYNCNT          48 PREG                87 BELL
 03D9 RWTS           03E3 LOCIOB        0578 RTRYCNT           133B ZNBUF2
 1341 ZNBUF1         1353 NOGOOD        1367 ENDERR            1376 DONETRK
 B8DC READ16         B944 RDADR16       BB00 NBUF1             BC00 NBUF2
 BF0D DSKF2          FDED COUT
```

use the DOS Sectors for user programs is to modify the VTOC to indicate them to be available for use. The same applies to the catalog sectors on track 17 ($11).

One interesting point hinted at above is that DOS does not use all the sectors in Tracks Ø, 1 and 2. In fact, most of Track 2 is unused; only sectors Ø to 4 (5 in all) are used and, therefore, should one want them, 2816 bytes are available on Track 2 without damaging the DOS at all and without affecting the BOOTability of a disk. This is useful if one uses a turnkey system with program disk and lookup data files in drive 1 and data disks (without the DOS on board) in other drives.

Using Track 35 for Storage

I first hit on the idea of using Track 35 ($23) when I noticed some protected disks use it to store information. Using Track 35 provides an extra 16 sectors or 4,Ø96 bytes (4k) to the user. Most Apple disk drives are quite capable of accessing Track 35; only the very early ones (pre DOS 3.2) have a little difficulty sometimes. I have used Track 35 on most of my disks for almost a year now and have experienced no problems with drive errors or alignment.

To use an uninitialised track, two things must be done. First the track must be initialised, that is, it must be formatted by writing to it an image of a blank track so the DOS can read it should it have to and, second, the VTOC must be amended to inform the DOS that those tracks are available for use.

There are basically two ways of getting onto Track 35:

1. Modify the DOS so it will automatically initialise Track 35 during INIT and update the VTOC accordingly. The disadvantages of doing this is that only blank disks can be initialised without losing data and so this method cannot be used on disks containing data without going through the laborious process of recopying data from disk to disk.
2. The other way is to write a program that selectively initialises Track 35 and then updates the VTOC accordingly. This will preserve any data on the disk.

Accessing VTOC

All the discussions above refer to modifying the VTOC. To do this one must be able to read the existing VTOC from the disk in question, modify it and then write it back again. Listing 1 is a small program, 86 bytes long, called simply T/S MOD that does just that, by setting up the appropriate parameters for the DOS and then calling RWTS (Read Write Track Sector – a DOS routine). It is written in machine code and takes its commands from three page 1 memory locations, namely $ØØ, $Ø1 and $Ø2:
$Ø2:This is set to 1 for a disk read and to 2 for a disk write.
$Ø1:This is set to the desired sector number ($Ø-$F).
$ØØ:This is set to the desired track number ($Ø-$23).

On entry to the program all registers are saved, after which a jump is made to the page 3 memory location LOCIOB which contains a jump instruction to the DOS routine which locates the RWTS IOB (Input/Output Control Block). The reason for this step is to make the program memory size independent as it will locate the IOB no matter where it is.

The next section of the program proceeds to set up the IOB with the relevant Track and Sector data before proceeding to the final section where the DOS RWTS is called to read or write the sector to or from memory. If an error occurs then an error code is stored in memory location $Ø2 for the calling program to access and do something about. Before exiting, all registers are restored to their pre-entry states.

Initialising Track 35

Listing 2 is a delightfully short BASIC program called DOS 35 that will modify the DOS in memory so when a new disk is initialised using the DOS INIT function it will automatically be formatted to have 35 tracks. The memory locations apply to a 48K Apple, only. For a 32k Apple, subtract 16384; for a 16k Apple, subtract 32768 from the values given. Should the SPACE program below (qv) be used on a disk initialised with this DOS, no harm will come to the contents of track 35. The functions of releasing

Program listing 2 DOS 305

```
10 REM   DOS 35 : A Basic Program to modify a NORMAL DOS
                  so that it will create 36 track Diskettes
                  when the INIT command is used.


20 REM   (C) 1983.   by Pieter Meiring B.Sc.(Hons) M.B Ch.B.


100 POKE 44725, 144                        ; Location #AEB5 set to #90.
110 POKE 48894, 36                         ; Location #BEFE set to #24.

200 HOME
210 VTAB 10 : PRINT" Please Write or load your 'HELLO'
                     program and INIT a BLANK diskette
                     to obtain a 36 track diskette."
999 NEW
```

Program listing 3 T/S MOD Track and Sector Accessing Program

```
----- NEXT OBJECT FILE NAME IS T/S MOD.OBJ0
1400:                    ORG  #1400

                   *****************************************************
1400:              * T/S MOD: A Program to read/write any sector on disk.*
                   *****************************************************

1400:              *** ZERO PAGE USAGE

0003:              PTR     EQU  #3        Work Pointer
0002:              OPER    EQU  #2        Operation to be performed
0001:              SECTOR  EQU  #1
0000:              TRACK   EQU  #0
0001:              READ    EQU  1         Read operation
0002:              WRITE   EQU  2         Write operation
0048:              PREG    EQU  #48       MONITER P Register Save Area
0087:              BELL    EQU  #87

1400:              *** OTHER ADDRESSES

1770:              BUFFER  EQU  6000      Sector data buffer
03E3:              LOCIOB  EQU  #3E3      Locate RWTS Parmlist Subroutine
03D9:              RWTS    EQU  #3D9      RWTS Routine
FDED:              COUT    EQU  #FDED     COUT

1400:              *** RWTS IOB DEFINITION

0000:                      DSECT
0000:              IOBIOB  DS   1         IOB Type (#01)
0001:              IOBSLT  DS   1         SLOT*16
0002:              IOBDRV  DS   1         DRIVE
0003:              IOBVOL  DS   1         VOLUME
0004:              IOBTRK  DS   1         TRACK
0005:              IOBSEC  DS   1         SECTOR
0006:              IOBDCT  DS   2         Address of DCT
0008:              IOBBUF  DS   2         Address of Buffer
000A:              IOBSIZ  DS   2         Sector Size
000C:              IOBCMD  DS   1         Command Mode

000D:              *** COMMAND MODE CODES

000D:              * #00 EQU  NULL Command
000D:              * #01 EQU  READ Command
000D:              * #02 EQU  WRITE Command
000D:              * #04 EQU  FORMAT Command
000D:              IOBRCD  DS   1         Return Code

000E:              *** RETURN CODES

000E:              * #00 EQU  All well
000E:              * #10 EQU  Write Protected
000E:              * #20 EQU  Volume Mismatch
000E:              * #40 EQU  Drive Error
000E:              * #80 EQU  Read Error

000E:              IOBTVL  DS   1         True Volume
000F:              IOBPSL  DS   1         Previous slot
0010:              IOBPDR  DS   1         Previous Drive

1400:                      DEND
```

```
1400:48          68           PHA
1401:98          69           TYA
1402:48          70           PHA
1403:8A          71           TXA
1404:48          72           PHA

1405:            74 *** FILL IN RWTS IOB

1405:20 E3 03    76           JSR   LOCIOB     Locate RWTS Parmlist
1408:84 03       77           STY   PTR        and save pointer
140A:85 04       78           STA   PTR+1

140C:A5 00       80           LDA   TRACK      get TRACK to READ/WRITE
140E:A0 04       81           LDY   #IOBTRK    and store in RWTS list
1410:91 03       82           STA   (PTR),Y
1412:A5 01       83           LDA   SECTOR     Get Sector to Read
1414:C9 10       84           CMP   #16        Bigger than 16?
1416:90 04       85           BCC   SOK        NO

1418:A9 00       87           LDA   #0         Set it Back to 0
141A:85 01       88           STA   SECTOR

141C:A0 05       90 SOK       LDY   #IOBSEC
141E:91 03       91           STA   (PTR),Y    Store in RWTS List
1420:A0 08       92           LDY   #IOBBUF
1422:A9 70       93           LDA   #>BUFFER   Store Buffer pointer in List
1424:91 03       94           STA   (PTR),Y
1426:C8          95           INY
1427:A9 17       96           LDA   #<BUFFER
1429:91 03       97           STA   (PTR),Y
142B:A5 02       101          LDA   OPER       Get Command Mode
142D:A0 0C       102          LDY   #IOBCMD    and store in List
142F:91 03       103          STA   (PTR),Y


1431:A9 00       106          LDA   #0         Any Volume will do
1433:A0 03       107          LDY   #IOBVOL
1435:91 03       108          STA   (PTR),Y

1437:            110 *** NOW CALL RWTS TO READ/WRITE THE SECTOR

1437:20 E3 03    112          JSR   LOCIOB     Reload pointers
143A:20 D9 03    113          JSR   RWTS
143D:A9 00       114          LDA   #0
143F:85 48       115          STA   PREG       Fix P reg so DOS is Happy
1441:85 02       116          STA   OPER       Store OK code for Calling Program
1443:90 0B       117          BCC   EXIT       All is Well

1445:            119 *** ERROR OCCURED!

1445:A9 87       121          LDA   #BELL      ;BEEP the speaker
1447:20 ED FD    122          JSR   COUT

144A:A0 0D       124          LDY   #IOBRCD
144C:B1 03       125          LDA   (PTR),Y    Get RWTS Return Code
144E:85 02       126          STA   OPER       Store Error Code for Call Program

1450:            128 *** EXIT TO CALLER AFTER RESTORING ALL REGISTERS

1450:68          130 EXIT     PLA
1451:AA          131          TAX
1452:68          132          PLA
1453:A8          133          TAY
1454:68          134          PLA
1455:60          135          RTS

*** SUCCESSFUL ASSEMBLY: NO ERRORS

SYMBOL TABLE    SORTED BY SYMBOL                     05-05-1983 #00019 PAGE 4

    87 BELL          1770 BUFFER          FDED COUT            1450 EXIT
    08 IOBBUF          0C IOBCMD       ?    06 IOBDCT       ?    02 IOBDRV
?   00 IOBIOB      ?   10 IOBPDR       ?    0F IOBPSL            0D IOBRCD
    05 IOBSEC      ?   0A IOBSIZ       ?    01 IOBSLT            04 IOBTRK
?   0E IOBTVL          03 IOBVOL          03E3 LOCIOB            02 OPER
    48 PREG            03 PTR          ?    01 READ            03D9 RWTS
    01 SECTOR        141C SOK               00 TRACK        ?    02 WRITE

SYMBOL TABLE    SORTED BY ADDRESS                    05-05-1983 #00019 PAGE 5

    00 TRACK       ?   00 IOBIOB       ?    01 READ         ?    01 IOBSLT
    01 SECTOR          02 OPER         ?    02 IOBDRV       ?    02 WRITE
    03 PTR             03 IOBVOL            04 IOBTRK            05 IOBSEC
?   06 IOBDCT          08 IOBBUF       ?    0A IOBSIZ            0C IOBCMD
    0D IOBRCD      ?   0E IOBTVL       ?    0F IOBPSL       ?    10 IOBPDR
    48 PREG            87 BELL            03D9 RWTS            03E3 LOCIOB
  141C SOK           1450 EXIT            1770 BUFFER          FDED COUT
```

DOS and catalog sectors are, needless to say, not available in this program! For the person who wants least hassle and is satisfied with only 4k more storage this program is best.

Listing 3 is a short machine language program, 131 bytes long, called FORMAT that will format any track on the disk. It uses the same page 1 memory locations as T/S MOD above for user control.

Briefly, it works as follows:

On entry, all registers are saved on the Program Stack (Page 1). As with T/S MOD a jump is made to LOCRPL and the RWTS IOB is set up. The DOS RWTS is then called with a null command to position the disk drive head over the appropriate track. Parameters are then set up for the DOS Disk Format routine (DISKF2) which is then called to write a formatted track. After formatting, a read of the track is performed to test the track for integrity. Should an error occur, an error code is generated and stored for the calling program to sort out. Before returning control to the calling program, all registers are again restored.

It should be noted this program is memory dependant, and should it be written for systems of less than 48k RAM, the values of NBUF1, NBUF2, READ16, RDADR16 and DISKF2 should be amended to suit the amount of memory available.

Putting it all together

The machine language programs use a section of memory from 4864 to 6255 ($13ØØ to $186F). FORMAT resides in memory between locations 4864 and 5ØØ3 ($13ØØ to $1394). T/S MOD resides between memory locations 512Ø and 5196 ($14ØØ and $144B). These memory locations were chosen arbitrarily so T/S MOD could reside just above FORMAT, which is just above a BASIC program written to access them. 6ØØØ was just a convenient decimal number at which to locate the buffer to facilitate calculation of PEEK and POKE addresses to modify the buffer.

A sample BASIC driving program, called SPACE, using both T/S MOD and FORMAT is given in Listing 4.

It works as follows:

Lines 1ØØ-195: The machine code programs described above are loaded into memory and various variables initialised. Then the menu is encountered which asks what kind of modification(s) is/are (more than one option can be selected) required. On pressing (Return) execution of any chosen option commences by checking the option flags (C(x)) at the beginning of each routine.

Lines 2ØØ-29Ø: Deleting the DOS: The VTOC is read into the buffer by initialising page Ø locations for track 17, sector Ø and CALLing T/S MOD, checked it has not previously been modified and modified to free all the DOS sectors. Should the unused DOS sectors have been released previously, only those still occupied by the DOS image will be released. Upon modification, one of the unused bytes in the VTOC is used as a flag to indicate modifications so data is not lost should you try to modify the same disk again. The flag F1% is then set to indicate that the simple BOOT program (qv) is to be installed later.

Lines 3ØØ-39Ø: Freeing unused DOS sectors: As above but only the last 11 sectors in track 2 are freed. Again a flag is inserted for future mis-management trapping!

Lines 4ØØ-5ØØ: Initialising track 35 ($23): The VTOC is checked to see whether or not Track 35 is in use by looking at byte $34; the value of which indicates the number of tracks on the disk. If equal to 34 ($23) then memory location Ø is POKEd with 35 and FORMAT CALLed to format Track 35. The buffer is then modified at byte $35 and at the bit map of track 35 (bytes $C4 to $C7).

Lines 5ØØ-57Ø: Freeing catalog: The buffer image of the VTOC is checked for previous modifications and modified to indicate that sectors 1-14 ($1 to $E) on Track 17 ($11) are unused. Flag F2% is set to indicate that Track 17, sector 15 will need modification.

Lines 6ØØ-67Ø: The VTOC is written back to its place in sector Ø, track 17 from the buffer. If Flag F2% is set then Track 17 ($11), Sector 15 ($F) is then modified and the pointer bytes to the next catalog track and sector are set to Ø to indicate that this is the last catalog sector. This will restrict the maximum number of file names on the disk to 7. Other values that can be used to POKE into the buffer so that more file names can be

Program listing 4

```
                SPACE: More disk space under DOS 3.3      Page 2


0  LOMEM: 6256                                    ; Set to protect programs.

10 REM   SPACE: A BASIC program using the T/S MOD and FORMAT
               machine code routines to create more Disk
               storage space under the Apple ][ DOS 3.3.

; Please note that this is a dangerous program if used inexpertly. Data loss
; can occur especially when catalog sectors are released should there be more
; than 7 files createo on the diskette. Note that DELETED files also take up
; catalog sector space and may occupy sector $F! The best procedure is to apply
; the modifications to a BLANK diskette and then to create files on it or copy
; files onto it.

20 REM   (C) 1983.   by Pieter Meiring B.Sc.(Hons) M.B Ch.B.  (v1)

50 HOME : VTAB10 : PRINT"  SPACE : More Disk Space for the Apple."

99 REM Initialise Constants.

100 D$ = CHR$(4) : IN% = 4864 : TS%= 5120 : R% = 1 : W% = 2 : I% = 2
110 TR% = 0 : SE% = 1 : C% = 2 : NT% = 35 : V% = 254

119 REM Load in Machine code Programs.

120 PRINT D$ "BLOAD T/S MOD.OBJ0" : PRINT D$ "BLOAD FORMAT.OBJ0"

129 REM Print the Menu.

130 HOME : PRINT"              SPACE:
                               -----


                  Please choose your options by number:

                      1....Delete DOS from Tracks 0-2.

                      2....Free unused DOS sectors.

                      3....Free Catalog sectors.

                      4....Initialise Track 35.

                      5....Restart Program.

                      6....Quit.

                SPACE: More disk space under DOS 3.3      Page 3
137 C( 1 ) = 0 : C( 2 ) = 0 : C( 3 ) = 0 : C( 4 ) = 0

139 REM Get desired choices.

140 VTAB 21 : HTAB 6 : GET I$
150 I = VAL( I$ ) : IF I < 1 OR I   5 THE 140    ; Invalid Input rejected.
160 IF I = 5 THEN 130                            ; Start Over.
165 IF I = 6 THEN HOME: END                      ; Quit.
170 IF I$ = CHR$( 13 ) 200                       ; Start Working.
180 IF I = 1 AND C( 2 ) THEN 140                 ; Choices are incompatible.
185 IF I = 2 AND C( 1 ) THEN 140                 ; And again!
190 VTAB I * 2 + 6 : HTAB 1 : ?"->"              ; Mark Choice on Menu.
195 C( I ) = 1 : GOTO 140                        ; Record Choice.

199 REM Free all DOS sectors routine.

200 F1% = 0 : F2% = 0 : HOME                     ; Initialise flags.
205 POKE TR%, 17 : POKE SE%, 0                   ; VTOC location (trk 17,sec 0)
210 POKE C%, R% : CALL TS%                       ; Retrieve it.
220 IF NOT C( 1 ) THEN 300                       ; Skip section if not chosen
225 IF NOT PEEK( 6047 ) THEN 230                 ; Check for DOS DELETED marker
227 PRINT"
                                                 ; ERROR message if needed
         DOS has already been deleted!"          ; Control-J is used here.
228 GOTO 400                                     ; Since there is no DOS the
                                                 ; section can be skipped!
230 POKE 6060, 255 : POKE 6061, 255              ; Free Track 1.
```

```
240 IF NOT PEEK( 6046 ) THEN 260                    ; Check if DOS Track 2 has
                                                    ; been Partially freed.
250 X = PEEK( 6065 ) + 31 : POKE 6065, X            ; Free only sectors 0 to 4
255 GOTO 270
260 POKE 6064, 255 : POKE 6065, 255                 ; Free Track 2.
270 POKE 6047, 255                                  ; DOS DELETED marker.
280 PRINT"

          Deleting DOS and releasing sectors"
290 F1% = 1 : GOTO400                               ; Set Flag 1.

299 REM Free unused DOS sectors Routine.

300 IF NOT C( 2 ) THEN 400                          ; Skip section if not wanted.
310 IF PEEK( 6047 ) THEN 227                        ; Check if DOS is no more!
320 IF NOT PEEK( 6046 ) THEN 350                    ; Check if DOS partially freed
330 PRINT"

         Unused DOS sectors have already been freed!"
340 GOTO 500
350 POKE 6064, 255 : POKE 6065, 224                 ; Free unused Trk 2 DOS sectors
360 POKE 6046, 255                                  ; And update VTOC Flag
370 PRINT "

          Freeing unused DOS sectors"

399 REM Initialise Track 35 routine.

400 IF NOT C( 4   THEN 500                          ; Skip section if not wanted.
410 IF PEEK( 6052 ) = 35 THEN 440                   ; Check Number of tracks.
420 PRINT"

         Track 35 already in use!"                  ; ERROR message if needed.
430 GOTO 500
440 POKE TR%, NT% : POKE SE%, V%                    ; Setup to INIT track 35.
450 PRINT "

          Initialising Track 35."
460 CALL IN% : IF PEEK( C% ) THEN 700               ; INIT and branch on error.
470 POKE 6052, 36 .                                 ; Update no of tracks in VTOC
480 POKE 6196, 255 : POKE 6197, 255                 ; Free Trk 35 in VTOC.
490 PRINT "

          Track 35 initialised."

499 REM Free catalog track sectors routine.

500 IF NOT C( 3 ) THEN 600                          ; Skip section if not wanted.
510 IF NOT PEEK ( 6045 ) THEN 540                   ; Check if Cat track in use.
520 PRINT"

         Catalog Track already in use!"             ; ERROR message if needed.
530 GOTO 600
540 POKE 6124, 127 : POKE 6125, 254                 ; Free sectors $1to$E on Trk17
550 POKE 6045, 255                                  ; Set Flag in VTOC.
560 PRINT "

          Freeing Catalog track\

570 F1% = 2                                         ; Set Flag 2.

599 REM Reads and Writes.

600 POKE TR%, 17 : POKE SE%, 0 : POKE C%, W%        ; Prepare to write VTOC.
610 CALL TS% : IF PEEK( C% ) THEN 700               ; Write & branch if ERROR.
620 IF NOT F2% THEN 710                             ; To Menu if not Option 3.
630 POKE SE%, 15 : POKE C%, R%                      ; Setup to read catalog sect
640 CALL TS% : IF PEEK( C% ) THEN 700
650 POKE 6001, 0 : POKE 6002, 0                     ; Delete link to next sector.
660 POKE C%, W% : CALL TS%                          ; Write it back again.
670 IF NOT PEEK( C% ) THEN 130
699 REM Error Handling Section.

700 PRINT "

          DISK ACCESS ERROR!!!"
710 IF NOT F1% THEN 800                             ; Flag for deleted DOS.
```

listing continues

traded for less space are given in the comment lines.

Lines 7ØØ-71Ø: This is where all errors in the machine code programs cause the program to jump to keep the user informed about the error status of the operation.

Lines 72Ø-81Ø: This is a section of code that installs a BOOT ERROR message into track Ø sector Ø of disks where the DOS image has been deleted (flag F1% set). This measure is to inform the user he will not be able to BOOT DOS with the disk should he try to do so, and also to stop the drive motor from wearing out should a data disk have been left in the drive and power supplied to the computer with no-one in attendance. On attempted BOOTing, the computer will display an error message and stop the drive. The appropriate code and message is POKEd into the buffer memory after which the buffer is written to the disk.

Otherwise the programs are well annotated and self explanatory. These programs work and can be used as they are. Some special editing techniques were used in the BASIC programs to make them appear more readable but, except perhaps for line 13Ø (Menu) in SPACE, they can be copied without the semicolon comment lines and will work. If all the comments are desired, the semicolons can be replaced with :REM statements which will work with standard Applesoft.

Miscellaneous

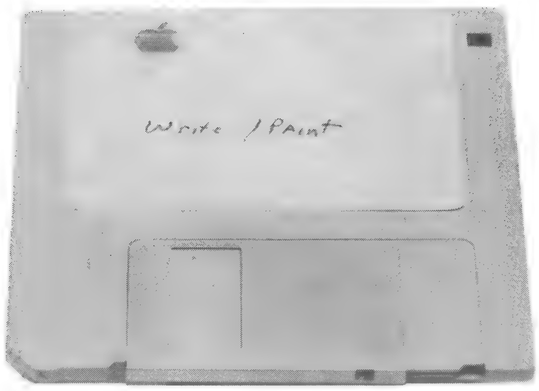
It should be noted that modifying the DOS, the catalog sectors and the VTOC may cause incompatibilities with certain commercial disk utility programs. These are basically programs that copy disks, fix up the VTOC if it should be damaged and provide other functions such as telling how much space is left in the disk, etc. Most copy programs work very well with SPACE disks.

Programs known to cause minor problems are:

Bag of Tricks: This excellent set of programs assumes a standard disk format and refuses to function beyond track 34. The Fixcat program will restore the disk to its standard DOS configuration; data in Tracks 17 and 35 will be lost if this is used, as

Disk drives in the future

by Duncan McCann



Once you have used your computer to work on a program you have to save it. When Apple first started, information had to be saved laboriously to a cassette tape recorder. Take it from one who suffered that it was not easy.

It is interesting that cassette storage is still the major means of saving information from computers in Australia, but not on Apples.

Almost immediately after the launch of the Apple II came the Disk II floppy disk drive. This was a Shugart drive, and when it was introduced by Apple in 1978 it swept all before it.

It was designed by Steve Wozniak, co-founder of Apple, and it is considered by many to be one of the great engineering accomplishments in personal computers.

Wozniak designed a revolutionary disk drive controller and the electronic circuit necessary to operate the disk drive during the week between Christmas and New Year's Day in 1977. His disk controller was much less expensive and far more reliable than anything else that had been developed.

Within three months of Wozniak designing the disk controller, Apple became the world's largest producer of 5.25 inch (13.2 cm) floppy disk drives.

Storage

Program listing 4 continued

```
719 REM DOSless BOOT error message instalation Section.

720 RESTORE
730 PRINT "

          Installing BOOT ERROR message"
740 S$ = "This Diskette has no DOS on this side." ; DOSless BOOT ERROR
745 M$ = CHR$( 13 ) : S$ = S$ + M$ + M$
750 S$ = S$ + "Please Re-BOOT using another one."
755 FOR I = 6000 TO 6042
760 READ P% : POKE I, P% : NEXT                     ; Install BOOT code in Buffer.
765 FOR I = 1 TO 70
770 POKE 6042 + I, ASC( MID$( S$, I, 1 )) : NEXT ;
775 POKE 6112,0
780 POKE TR%, 0 : POKE SE%, 0 : POKE C%, W%         ; Write Code to diskette
790 CALL TS% : IF PEEK( C% ) THEN 700               ; Track 0, Sector 0.

800 PRINT "
          Hit any key to continue."
810 GET I$ : GOTO 130
```

well as space (no data) in track 2 if DOS has been partially freed. In short these programs are to be avoided on SPACE disks.

Reconstruct VTOC: This program will rewrite the VTOC and all codes and flags put in by the SPACE program will be lost. Track 35 may also become hidden and data on Track 17 may be lost. To be avoided.

Super Copy: This program works exceptionally well and even displays the new additional track on its Track/Sector map. There are only two faults; it will not initialise that extra track when its Initialise function is selected and, should you use the Fix File Size option, all track 1, 2 and 17 sectors will become protected whether they have been used or not; data is not lost on any tracks. The disk space left displays are accurate and copying works well. (Oddly, tracks 1 and 2 are only protected if no program occupies them; Super Copy then probably assumes that the DOS image occupies them. If Fix File Sizes are used on a partially deleted DOS, free sectors on track 2 are lost although the data is not).

In conclusion, most programs I have used 35 track disks with have worked well with no problems, but it is well worth experimenting a little with your particular programs before entrusting valuable information to the cunning wiles of the monster computer. As with all things in life, it is well to back up everything well using something that is known to work before launching off into new ventures. □

Basic idea

Very basically, the floppy disk works from a flat circular plate of plastic coated with metallic oxide which is capable of holding a magnetic charge.

This circular plate is encased in a square plastic cover to protect it from dirty fingerprints, dust and cigarette ash. The square cover has a large circular hole in the centre which allows the diskette within to be rotated.

The Apple II diskette rotates above the read/write head at 3ØØ revs per minute. The head moves in and out over different concentric tracks of the disk.

Because they have limited the speed of rotation of the disk to a level where mechanical problems do not affect the rate at which data is stored, Apple floppy disk drives are a relatively slow method of storing permanent data.

On a standard Apple II disk drive there are 35 tracks and these tracks are about Ø.5 millimetres apart.

The read/write head uses only a portion of this width to allow some flexibility in the positioning of the disk and to allow for differences between drives.

Major developments

Since Wozniak devised the standard Apple II drive there have been major developments.

No matter how quickly memory prices fall, uses for the memory increase far more rapidly. AppleSoft Basic, which took over from Integer Basic on the original machine, took up more memory.

The new operating system ProDOS, which has taken over from DOS 3.3, takes up more memory.

Programs which once used to take up two or three K now require multiples of memory in order to operate properly. In the newer Apple machines this problem is even more acute.

The Macintosh in its original release had 128K of RAM, but many of the programs available for that machine fill up that space with ease.

As soon as the 256K RAM chips are available at the end of the year, the Macintosh is going to be operated so there is half a megabyte of memory available for programs.

Even the Lisa in its highest configuration, providing as it does a megabyte of memory, does not leave enough room for the entire operating system and common utilities to fit into the memory at the same time.

Holding pattern

The result of this is that one of the functions of a disk storage system is to hold sections of programs and information too big to fit into the computer's active memory.

How long this situation will continue depends on the speed with which the Random Access Memory on computers is expanded.

In this particular use the disk and the disk drive are frequently being made to act as if they were a Read Only Memory chip.

In this mode two aspects are absolutely vital. One is speed and the other is reliability.

Speed is important because the speed of the disk drive is very frequently the major factor retarding the speed of operation of a computer. If a delay in obtaining information lasts more than a second the user's attention starts to drift.

Reliability is obviously just as important.

Sadly, speed and reliability tend to be inversely related.

At Apple the decision has been that they will have reliability first and speed second.

The disk drives invented by Wozniak were roughly comparable in speed to other drives of the same time. But they are much slower than many of the drives that have been introduced in later years.

Slim drives

First of all, we have seen the slim line drives which have come out of Asia like Genghis Khan.

They provide the same amount of memory as a standard Apple disk drive but they take up less space.

Even more useful are a series of disk drives made by Rana. These are the Rana Elite series.

There are three versions: the Elite I, a 35/4Ø track drive; the Elite II, a double sided single density 8Ø track drive; the Elite III, a double sided double density 16Ø track drive. They are imported into Australia by CompuMusic.

To allow these drives to be used effectively Rana also produces a disk controller and a utility for modifying DOS to work with the drives.

All three Rana drives are very similar in styling to the Apple Disk II but are about a centimetre taller. All three versions use the same kind of door mechanism as the Apple Disk II – the door latch closes the drive, centres the disk, engages the hub and places the head above the disk.

Small advantage

With the Rana Elite I you have a single sided, single density 4Ø track drive which is almost precisely the same as the standard Apple I until you modify the DOS. Then instead of using 35 tracks you end up using 4Ø tracks, which increases the storage capacity by 2ØK. This is not a major memory advantage.

The Rana Elite II is similar to the Elite I, except it can write to both sides of the disk and therefore has 8Ø tracks. Since it is single density it can read and write to the ordinary 35 tracks as well.

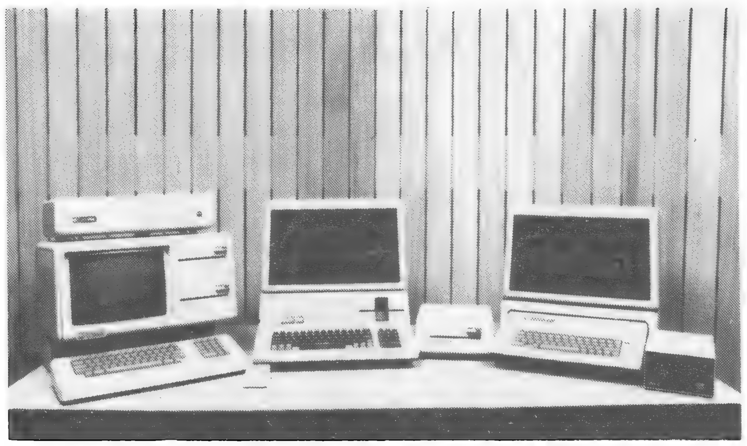
The result is a machine which gives you over double the amount of disk space available on the standard Apple drives.

Finally there is the Rana Elite III, which is both double sided and double density. It has 8Ø tracks on each side, a total of 16Ø tracks. In effect, this gives you four times the amount of memory area on a disk that you get on a standard Apple II drive.

To run it you need a Rana disk controller, which can control up to four drives when you are using the modified version of DOS. Without the modified DOS only two drives are accessible. The Rana disk controller can also control the standard Apple Disk II drive and its many clones.

The problem with these drives is, as always, the cost.

It is true that you can cram an immense amount of information onto the drives and it is true that they are extremely reliable. But in the end you have to look at the cost per kilobyte stored. And it may be that except in semi-professional situations the cost is too high.


Lisa with Profile hard disk drive (far left), Apple III with Unifile disk drive (middle) and the Apple IIe with the disk drive II.

Amazing developments

However, Rana is an amazing company which keeps on improving its products to keep up to the level of progress within the personal computer industry.

Soon to be available in Australia from CompuMusic is the quite staggering Rana 8Ø86/2 co-processor dual drive for the Apple //e which takes the Apple //e into a different area of personal computing altogether.

First of all, the two single unit drives can either operate as standard Apple II drives, each storing 143K of information. Or they can operate under ProDOS – Apple's relatively new operating system – and will then give 32ØK of storage space. Or, and this is where the system gets interesting, they will run under the IBM compatible MS-DOS as a pair of double sided double density drives.

When working under MS-DOS the drives work through an 8Ø86 18 bit microprocessor which makes the Apple think that it is an IBM clone.

Indeed, slightly better than that. Because the 8Ø86 works at higher speed than IBM's Intel 8Ø88 microprocessor. And built into the board is an incredible 256K of Random Access Memory which can be expanded to 512K.

Available software

Software is available to support all these amazing features and also to allow the use of CP/M and Pascal. The makers claim, and we think it is perfectly feasible, that it:

"allows data and text files to be written on other MS-DOS machines to be brought over to the 8Ø86/2, and to be read or written directly without modification. This lends a very high degree of portability to the user in business and professional environments."

Forget for the moment the American computerspeak involved in that statement and concentrate on the facts.

If they are true it may well be that once the whole system has been properly installed you will have an Apple //e that thinks it is an IBM and runs all the IBM programs – only faster. And still operates perfectly well as an Apple //e – but with superior disk drives.

We have not tested this system yet, but knowing the reputation of Rana we are fairly confident the whole package will work as advertised. Which will mean a minor revolution for Apple. For the first time you will be able to have your cake and eat it as well.

Microfloppies

We have seen microfloppies being produced in Taiwan which were double sided, double density and which cram one megabyte of information onto a 9 cm micro floppy disk. In Taiwan they were selling for less than $1ØØ each. Now that is what we call cheap, high capacity storage.

Quality is another matter. And we won't see those drives available in Australia until sometime next year, when they will be priced at $2ØØ.

Hard disks can spin their disks far more quickly than any floppy, micro or otherwise, and as a result they cut waiting time. They also allow faster data transfer as well.

The Apple II family was never originally designed to take hard disks, although the Lisa and the Macintosh were both created to work in this environment.

Apple, keeping to their tradition of safety before speed, have one of the slower hard disk drives on the market in the Apple ProFile.

The technology of the hard disk is very similar to that of a floppy disk but there are radical differences:

1) The disks instead of being made of mylar or some other plastic are normally made of aluminium alloys.

2) These platters have extremely thin coatings of iron oxide to take the magnetic charge. A coating that is far thinner than on the standard floppy disks.

3) The disk operates at about 3,6Ø Ø rpm versus about 3ØØ rpm for floppy disk drives.

4) The read/write head never actually comes in touch with the rapidly rotating disk. It flies on a very thin cushion of air, and never comes to rest on the information area of the hard disk.

5) The read/write heads rest in a specially prepared landing zone.

6) The whole of the storage disk, the read/write heads and the positioning mechanism are enclosed in a sealed environment so that dust, dirt and other polluting elements cannot enter.

Significant increase

Working on the basis that five

megabytes is the equivalent of about 35 diskettes, you can see it is a significant increase in memory. It is also a significant increase in speed of access because the information is collected in hundredths of milliseconds as opposed to milliseconds.

Although initially hard disk drives can seem extremely expensive, if you make a comparison with the equivalent number of floppy disk drives they work out relatively cheap.

The main problem with hard disk drives is backing up the information.

You can either do selective backups, which means that you record any important files that have been changed since last time they were backed up.

Or you can use an external cartridge tape drive which will allow the whole of the hard disk content to be recorded on removable tape.

In the past year or so several companies have been announcing removable cartridge hard disk drives. These drives offer the advantages of a hard disk plus the convenience of the removability of floppies. At the moment they are extremely expensive, but there is no doubt that in the near future we will be seeing this option offered at a reasonable price.

The theory behind the hard disk design is that because the read/write head is coming extremely close to the disk's coating at high speed it decreases the time the head needs to find information. At the same time it allows for a density of about 24 tracks per millimetre, compared with something between 2 and 4 tracks per millimetre on a floppy disk drive.

This means that you can measure the times for seeking information on a hard disk in milliseconds instead of hundreds of milliseconds on a floppy disk.

High transfer rate

At the same time hard disks give a data transfer rate of megabytes per second, which makes hard disks relatively efficient in operations where disks are continuously being accessed.

However there is a down side to the situation.

Because of the close mechanical tolerances involved everything must be kept absolutely spotless, otherwise you get a head crash.

When that happens the data get zapped, and zapped permanently.

The problem was much more serious with early hard disk technology, but nowadays disk drives are so well sealed that problems are relatively rare.

However, the use of a back-up is still recommended at all times.

Another way around the speed of access problem is to use a RAM disk emulator board, which may one day incorporate semi-permament memory.

There is still much to be done with the ability of the machine to save data and programs once you turn its power off.

Apple computers of course have some built in permanent memory in their Read Only Memory, but the Random Access Memory in the computer loses its contents the moment you turn off the power.

There is no doubt this is the area where there will be changes in the near future, as CMOS technology allows RAM chips to retain their memory.

No moving parts

The computer accepts the fact that this memory is a disk drive but because there is no mechanical operation involved the wait for information to be retrieved is invisible to the human eye.

By using disk memory instead of the main memory, RAM pseudo disks can use existing software that expects to access information on a continuing basis from disks. Whether you use them as memory extension for programs or whether you use them as a place in which to load the operating system of the program, or indeed, both, the fake RAM disks give an outstanding combination of reliability and speed.

Delays are almost negligible and the transfer rate is usually as fast as the machine's ability to accept information. As we get bigger and bigger Random Access Memory chips, so we are going to get more and more Random Access Memory pseudo disk systems.

You have excellent reliability because on a RAM pseudo disk there are absolutely no moving parts. The data and information is only read in once. After that you can read and write data as frequently as you like and it will not wear out.

Speed and reliability

Equally, a RAM disk board will never lose a bit of data unless you have a complete chip failure or some other hardware problems.

After speed and reliability comes capacity.

Just as software is continuing to overflow available main memory, the amount of data that can be stored on disk grows ever larger.

One of the major considerations is how much information you can access at any time.

The amount of data information that a disk drive can store depends on the design of the disk drive and how the information is encoded into bit patterns.

Again there is a balance between reliability and density.

Denser storage allows less room for slight errors and is thus inherently less reliable.

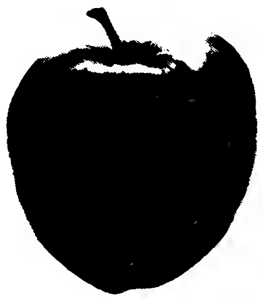
At this moment nobody is actually sure in which direction mass storage for Apples is going to go.

There is no doubt that a higher capacity floppy disk drive for the Apple II family has been overdue for some time, and even the 1Ø megabyte hard disk drive on the Lisa has had to be extended towards the 35 megabyte range.

The future

The probability is that we are going to see some of the high capacity 13 cm floppy disks and 9 cm micro floppy disks as standard with the Apple II family in the very near future.

Probably we will see a lot more RAM disks when the 256K RAM chips are available. We may see hard disks built into the machine with capacities in excess of 5Ø megabytes. That is our guess. Your guess is probably as good as ours. □



The first Computer Of The Year that won't be out of date by next year.

The highly respected "Your Computer" magazine has just named the Apple Lisa® as Computer Of The Year, 1984.

In their own words, "People will remember 1983 as the year that Lisa revolutionised personal computing."

Surely good news for the business on the verge of choosing the ideal system.

In the frantic, fast-moving world of micro

technology, where new models are here today and gone this afternoon, Lisa seems to be a reassuring exception.

This is the most advanced personal computer in the world, with up to one million bytes of internal memory.

Unlike conventional computers, Lisa works visually, the way you do. Those complex computer commands are replaced with familiar symbols and a palm-sized mouse.

Countless man-hours are saved because Lisa starts being productive from the moment it's switched on. (Even for staff who've never used a computer before.)

Little technical miracles like these don't exactly happen overnight.

Considering they've taken us a good five years to perfect, even if our competitors simply copy, they should be kept busy for some time.

There are three Lisa models of varying price and capacity, any one of which your Apple dealer would be proud to demonstrate.

You probably won't be the only company moving to Lisa technology this year.

We expect quite a few of our competitors will be doing likewise.

apple®

Lisa. The personal computer for the office.

Number Farm/Rodeo/Circus

Number Farm

Now available for the Apple II, II+ and //e microcomputers is an educational game for children up to seven years of age. Called Number Farm, the game helps parents to teach their children how to count. Parents are an integral part of this p]rogram. Put a child in front of the computer with this disk running and he will not know what to do. He can not be expected to, because any child who is old enough to read the instructions for himself will not find this program of any use.

This program allows the child to sit on his father's lap following the directions which his father reads for him from the screen and interprets for him if necessary. The child can then learn something about the keyboard as he inputs the numbers the computer asks for.

A farmer usually stays on screen raising his arms or, on rare occasions, raising his hat to reveal a part running straight down the middle of his head.

The sound effects are good. The program plays music. One of the games features the barking of a dog and the mooing of a cow. As the farmer raises his hand to his ear to listen to the mooing the child can listen for the moos or simply watch to see how often the cow opens its mouth to moo. Then the child answers the question of how many times the cow mooed. If he gets the number wrong then the farmer asks him again.

In the guessing game, the child is asked to guess how many eggs have been laid by the nine chickens on screen. When the child guesses he is told either his answer is too low, too high or is the correct number. This allows the child to exercise his mind in the use of numbers.

The child inputs the number by pressing the appropriate key on the keyboard. Thus the sequence in which the keys are arranged on the keyboard reinforces, in the child's mind, the concept of numbers being in a specific order.

To run the program you need an Apple II, II+ or //e with 48K memory, a disk drive and a, preferably colour, monitor or television screen.

Shape and Color Rodeo

This program is available for the Apple II, II+ and //e computers.

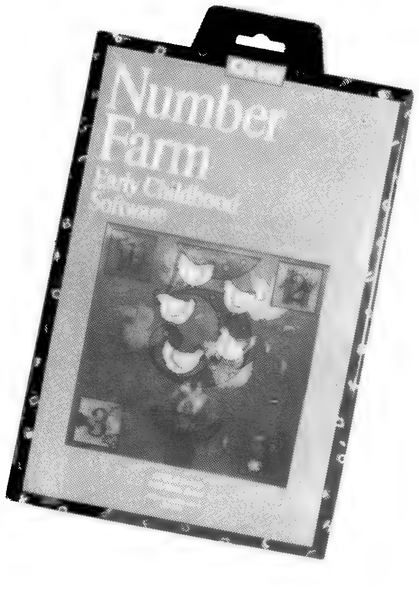
Designed for the under seven year old, this game displays various shapes and colours for the child to identify. And because the program teaches children to identify colours, a colour monitor and RGB card are essential. The instruction booklet tells you what is needed to run the program, as well as giving a rundown of each of the games.

The shapes used are a square, triangle, star, rectangle and two shapes which resemble a circle and an oval.

If you have a colour monitor, then this game should be helpful in teaching your child how to identifty colours. But, initially, it is you who has to name the colours. The instruction booklet also recommends the use of the cue card included and crayons.

Alphabet Circus

In this program the use of a colour monitor is not nearly as important because the program is designed to teach your child the alphabet. Having colour helps, of course, because it holds the child's attention.



The program starts with a tiger jumping through a hoop held by the master of ceremonies. He is present throughout the program to encourage your child as he learns to identify letters of the alphabet and arrange them in their conventional order. But you are also needed not only to read the instuctions but to tell the child what the letter sounds like. The program provides sound but only in the form of music and if you want the child to learn the names of the letters the computer will not speak them to him. You have to do that.

In one of the games, Marquee Maker, the child types a message onto the screen. It is printed in little letters half-way down the screen while it appears in big letters at the top of the screen. Pressing the space bar removes the word from the big lettered screen to make space for the next word. The big letter screen will take a maximum of 12 characters so your child will not be able to type in "schizomycetes". The small letter screen is restricted to 37 characters so the child will not be able to type the "To be or not to be" speech from Hamlet.

But if you want all that then you want a word processing program and this program was not designed for anyone that skilled.

All three of these programs were designed for very young children. So they are not designed for people who can be left to work the computer on their own; using them is going to require a lot of time and patience of the parent's part. The trick is to treat the learning process as a game and these programs do that with interesting graphics and music.

These programs are only as good as parents can make them. But if the parent is willing to persevere and is willing to play around with pens, paper and crayons then these programs can be a help in teaching kids the basics of the computer keyboard and number system.

The programs are produced by DLM, under the Early Childhood Software label and distributed by Dataflow. They retail at $39.95 each.

A Pot-pourri of Apple delights

by Derek Chan

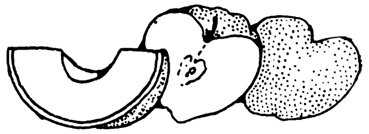
Apple Colours

Everyone knows that the Apple is capable of displaying 16 colours in the Low Resolution Graphics Mode (Lo-Res) and 6 colours in the High Resolution Graphics Mode (Hi-Res). Well, this is not strictly correct as the following short program will illustrate – tap it into your Apple and see for yourself!

```
 1Ø HOME : HGR
 2Ø FOR COL = Ø TO 255
 3Ø POKE 228,COL
 4Ø HPLOT Ø,Ø : CALL 62454
 5Ø VTAB 23
 6Ø PRINT "COLOUR CODE =
   ";COL:PRINT
 7Ø PRINT "HIT ANY KEY TO
   CONTINUE";
 8Ø GET K$ : HOME
 9Ø NEXT
1ØØ TEXT : HOME
```

The memory location 228 (hexidecimal $E4) contains the colour code for the colour specified by the APPLESOFT statement 'HCOLOR ='. For example, when HCOLOR = 1 (green) the value 42 will be stored in location 228 ($E4). In the FOR..NEXT loop we try out all possible values of the colour code Ø to 225 ($ØØ to $FF). Enjoy the colours, some of them are rather pretty.

In the APPLE ROM there are loads of high quality software or firmware (the Editor will surely make me do penance for using computerese) that good programmers should use. After all, why try to reinvent the wheel when it's been done for you for free. Line 4Ø is one such example. There is a MONITOR subroutine called BKGND with entry point 62454 which fills the Hi-Res screen with the last HPLOTted colour. This is considerably faster than trying to accomplish the same task with HPLOT statements. Do compare if you are sceptical.



HI-RES pages

The APPLESOFT Manual tells us that there are two High Resolution Graphics Pages. These can be invoked by the APPLESOFT commands HGR for Hi-Res Page 1 and HGR2 for Hi-Res Page 2. Internally APPLESOFT keeps track of which of the two Hi-Res Pages to draw on, for instance in response to a HPLOT statement, by storing an appropriate number in the location 23Ø ($E6). This memory location is called HPAG.

If graphics outputs are to be sent to Hi-Res Page 1 the content of HPAG will be the number 32 ($2Ø); if Hi-Res Page 2 is to be used HPAG will contain 64 ($4Ø). This is true irrespective of which Hi-Res Page is being displayed.

'Why try to reinvent the wheel when it's been done for you free'

The standard technique of producing smooth flicker-free graphics animation is to draw and display successive frames of the moving picture alternately on the two Hi-Res Pages. That is, we display a frame on Page 1 while drawing the next frame on Page 2 while the following frame is being composed on Page 1. This is repeated to give a smooth animation sequence.

The curious Apple devotee may ask: Where will the graphics output go if other values are stored in HPAG? The answer is, not surprisingly, to Hi-Res Pages 3 or 4 or 5. No, there have been no misprints!

The Apple Hi-Res Graphics system is Memory Mapped. This simply means the hardware circuitry has been constructed to display the numerical content of specially designated memory locations as dots on the video screen. Hi-Res Page 1 corresponds to the locations $2ØØØ- $3FFF; while Page 2 corresponds to $4ØØØ-$5FFF.

Now if the value 96 ($6Ø) is stored in HPAG, Hi-Res Page 3 ($6ØØØ-$7FFF) will be used for graphical outputs; the value 128 ($8Ø) will get us Page 4 ($8ØØØ-$9FFF) while 16Ø ($AØ) in HPAG will get us Page 5 ($AØØØ-$BFFF).

'We have effectively changed the value of the Y-Register'

WARNING: Drawing to Pages 4 and 5 will trample over DOS unless DOS has been moved to the language card area. With DOS active the highest available memory (HIMEM) is $96ØØ.

There is, however, no way for the Apple hardware to access Pages 3-5 directly. To display these pages, the content of these pages must be copied onto Pages 1 or 2 and then displayed as usual.

I will leave it to the curious reader to explore the effects of storing other values in HPAG. The hexidecimal values of the content of HPAG and the starting address of the Hi-Res Pages look interesting. I would strongly suggest saving any such exploratory progams BEFORE actually running them – otherwise the consequences may be blood curdling! Don't forget to set HIMEM accordingly. Write in and share your interesting experiences with these extra Hi-Res Pages.

'Experiences with these extra Hi-Res Pages'

Text/Graphics Soft Switches

These are not the results of short circuiting your Apple.

These are peculiar memory locations that have intimate links with the Apple hardware. By writing to these memory locations it is possible to make the Apple do interesting things.

The general way to use these soft switches in APPLESOFT is by the statement:

```
POKE SWITCH, Ø
```

The value of SWITCH depends on what we would like to do. What is being written to these memory locations or SOFT SWITCHES is irrelevant – so Ø is as good as any value. The TEXT/GRAPHICS SOFT SWITCHES are the eight locations 49232-49239 ($CØ5Ø-$CØ57). These allow us to select between:

1. Text or Graphics Modes OR Display.
2. Full screen graphics OR Mixed Graphics and Text
3. Display Page 1 OR Page 2. These may be Page 1 or 2 of the Graphics or Text screen depending on whether Graphics or Text Mode is being selected.
4. Hi-Res OR Lo-Res Graphics.

'In a family computing magazine I am not permitted to be more explicit'

The value of the switches are as follows.

Effect	Switch
Graphics Mode	49232
Text Mode	49233
Full Screen Graphics	49234
Mixed Text/Graphics	49235
Page 1	49236
Page 2	49237
Lo-Res Graphics	49238
Hi-Res Graphics	49239

For example, to select Full Screen Hi-Res Graphics Page 1 we execute the program segment:

```
POKE 49232,Ø        Graphics Mode
POKE 49234,Ø          Full Screen
POKE 49236,Ø               Page 1
POKE 49239,Ø               Hi-Res
```

Note the Text/Graphics Mode switch should always be selected first. For the other switches, once selected they do not have to be re-selected unless changes are desired. For instance, in Graphics Animations we may want to display Page 1(2) while we draw on Page 2(1). This can be accomplished by the pair of statements:

```
POKE 23Ø,32        Draw on Page 1
POKE 49237,Ø       Display Page 2

POKE 23Ø,64        Draw on Page 2
POKE 49236,Ø       Display Page 1
```

'Smooth flicker-free graphics animation'

Moving Memory

This is not a reminiscence about an encounter with Robert Redford or Bo Derek or Boy George – depending on your predelictions. In a family computing magazine I am not permitted to be more explicit.

On occasions you may like to move the contents of a block of memory to another location. For example, you may want to move a picture you have drawn on Hi-Res Page 3 to Page 2 to be displayed. This can, of course, be done by PEEKs and POKEs as follows:

```
For I = START TO FINISH
POKE (DEST+START-I) , PEEK (I)
NEXT
```

where the source data is in between memory locations START and FINISH; and DEST is the starting address of the destination location. Unfortunately this method tends to be unacceptably slow.

An alternative is to use the subroutine in the MONITOR in ROM. This subroutine is called MOVE – entry point –468 ($FE2C). In fact, MOVE is a misnomer. The content of a specified block of memory is COPIED to another location; but the original memory remains unaltered.

In order to use MOVE we need to specify the various addresses:

1. The two bytes STARTing address must be stored in locations 6Ø and 61 in the usual low byte first convention. The locations 6Ø and 61 are referred to as A1 in the official Apple literature.
2. The FINISHing address must be in locations 62 and 63 which are known as A2.
3. The DESTination address must be in locations 66 and 67, known as A4.
4. The Y-Register of the 65Ø2 Central Processor Unit (CPU) must be set to Ø.

The above data can be set up with the program segment:

```
1 POKE 6Ø, START - INT(START/
256)*256:
POKE 61, INT (START/256)

2 POKE 62, FINISH - INT(FINISH/
256)*256:
POKE 63, INT (FINISH/256)

3 POKE 66, DEST - INT (DEST/
256)*256:
POKE 67, INT (DEST/256)

4 CALL-182: POKE 71,Ø: CALL-193

5 CALL-468: REM Call MOVE
```

Line 4 has the effect of storing a Ø in the Y-Register of the 65Ø2 CPU. Since there are no direct ways of storing values in the Y-Register in APPLESOFT, we call on two other subroutines in the MONITOR ROM. The first one is called SAVE (-182) which SAVEs all the contents of all the internal registers of the CPU in predetermined memory locations in RAM. In particular, the content of the Y-Register is SAVEd at location 71. The other subroutine we used in line 4 is RESTORE (-193), which as the name suggests does the exact opposite of SAVE. Thus, by changing the value of location 71 between a call to SAVE and a call to RESTORE, we have effectively changed the value of the Y-Register.

If you haven't noticed, you have just been manipulating the CORE OF THE APPLE – that is the 65Ø2 CPU directly. It hasn't been too painful, has it? □

Brian Hammerhead is the "Hell fire Warrior"

by Alan Fenton

Hardware:	48K and 1 disk
Language:	Applesoft / Assembly
Company:	Automated Simulations
Graphics:	Hi-Res

The adventure game Hellfire Warrior consists of levels 5 to 8 from The Temple of Apshai; however, Hellfire Warrior is an entire game on its own, so knowledge of the Temple of Apshai is not needed.

First we visit the Innkeeper, where we can make a character composed of the following; intelligence, intuition, ego, strength, constitution and dexterity. These attributes range from about 8 to 18 and help decide how your character will perform (strength determines how much weight can be carried).

About 25Ø gold Royals are supplied and a name of your choice. If you have a character from Temple of Apshai or want to make a super warrior, then the Innkeeper allows you to make your own character with weapons and armour. A stored character can also be loaded.

Next a visit to Gulik the armourer to haggle, for in the shops objects have a weight, quantity available and offered price. The price can be haggled over with the storekeepers, who argue briskly, and are not always complimentary. The armoury offers a sword, armour (from leather to full plate), shields of two sizes and bows with arrows.

Fnord's apothecary shop offers magic potions to boost your attributes. The effects usually last until you come back from your adventure; some are permanent. Which potion does what is a problem to amuse you for a while. I suggest using one potion at a time and recording the results, ie, +1 to ego, entering level 5 and returning. Go through the same procedure twice or maybe three times with each potion.

Malaclypse the mage (wizard) offers magic to add from +1 to +9 to your sword and/or armour, making it better for killing, magic-using or creating monsters. Magic objects are offered, like seven-league boots which let you run faster without tiring as easily, magic arrows and a few other objects.

The quest is to rescue Brynhild the warrior queen. She is sleeping in a deep cavern waiting to be brought into the sunlight, which will awaken her from her nine hundred year sleep.

'Which portion does what is a problem to amuse you for a while'

A story of Brian Hammerhead sets you in the right mood for the quest. Graphics are Hi-Res and in colour. On the screen on the left side a display like a house plan of the immediate area shows yourself (a person with a sword) and a monster. One way of mapping is to trace off the screen, a good use for the back of printer paper. The right upper corner displays room number (when applicable), wounds (get down to Ø and you're dead), fatigue (too tired and you can't move), weight of objects carried, arrows available, magic arrows available, name of monster attacking you, the number of your slain victims and a letter of the last desired action that your character performed. Movement is from the keyboard with different letters for the actions. A command summary card is supplied to prop before you.

Level 5 monsters mainly consist of giant insects. Level 6 is a maze with a hidden exit. Level 7 monsters (the undead variety) can make life very hazardous (running is a useful defence). A key has to be found to be used in level 8.

In level 8 the screen is mostly black – you are wandering in a vast cavern of hell. After getting a certain treasure you have to go in a certain direction to find the entrance to the final cave. At the entrance is a nine headed Hydra which can ruin your health something terrible. Run by it and "O"pen the door, but only if you have the key from level 7.

On the way a treasure consisting of gems and armour melted together (someone muffed it) should be left behind because it is too heavy to carry with Brynhild. One day I retrieved the treasure and left Brynhild behind and got a fortune from the Innkeeper.

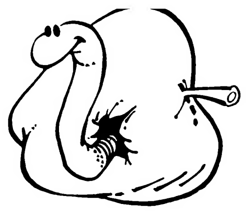
Treasures are found in different places, and when "G"et is used, a treasure number flashes on the screen which has a description for the appropriate level.

The game can be saved at any time or the character only after it has sold its treasure at the Inn.

Hellfire Warrior consists of one command summary card, one manual and one disk which may have Apple on one side and TRS-8Ø on the other. If you don't have any use for the other computer version, format it and store your characters on it.

'Brynhild is just another treasure number'

I have one reservation – the ending. Brynhild is just another treasure number, and all the treasures are converted to gold Royals at the Innkeeper's. Brynhild is worthless so maybe you should leave her to her beauty sleep, and "G"et the melted armour or maybe I will leave the choice to you. A very good game and if you like the style there are two expansion kits, (1) The Keys of Acheron and (2) Danger in Drindisti. □



Worm in the Apple

Will some kind gentleman at the stately pleasure dome in Ryde please explain to this lowly creature what, precisely, is happening to the Apple IIe? This worm was allowed sight of a letter that had emanated from the highest levels saying, in essence, that all Apple IIes were sold out and not to hold your breath as there wouldn't be any available until next February.

A probing journalist – are there any other kind? – for a newspaper that shall be nameless, "The Australian", telephoned and asked how could this be? How could Apple sell out of their best seller?

How, indeed?

Back came the strangest answer.

First it was pointed out that Australian Apples are made in Ireland. Secondly, the Irish had been on holiday. Third, there had been a shortage but the drought had been relieved by an airfreight shipment of some 1,600 machines. Fourth, Apple IIes with European configuration were now arriving in large quantities and another letter was being sent to retailers telling them to sell as many as they liked as stocks were more than adequate.

Which brings to mind a strange story.

Once upon a time the Duke of Wellington was standing on the steps of the Reform Club in London when a man approached him and said, "Mr Smith, I believe."

The Iron Duke raised an eyebrow and replied, "Sir, if you can believe that you can believe anything."

I don't know why I have just thought of that story. It just flashed into my mind.

Bits not bytes

It is not for a lowly worm to point the finger of shame at any senior executive at Apple. By and large they are a fairly decent bunch, even though they do make their own suits. But I really feel it would be helpful for me to point out that there is a difference between bits and bytes, and that the new chips in the Macintosh are 256K bit chips. Not 256K byte chips. Honest, guv'nor. I only mention it to be helpful.

Sam sung blue

If you have an Apple IIe and look with great care at the monitor, you will see it is built in South Korea by a company called Samsung. Personally, I think the monitors are the least well built part of the Apple computer, but it is not for a lowly worm to comment on matters like that.

However, my South Korean worm friend working off the island of Cheju (he's rather keen on the pearl divers there) tells me this amazing tale. Apparently an American company that makes a disk drive called Tandon have just taken considerable umbrage over the fact that their machines are apparently being pirated in South Korea. They have therefore slapped law suits on all and sundry and are claiming damages around the $100 million mark – give or take the odd dollar.

And named in the suit is a major Korean computer manufacturer. Although I have no musical talent I'm working on a song which goes "Sam Sung blue, everybody knows one . . ."

Who knows, it may make the Top Twenty.

Will the pop up pop up?

What then is happening with the pop-up screen for the Apple IIc? There is a worm in Cupertino – a statement that has been made oft before – who supplies me with information. And the word from my transpacific Deep Throat is that Apple are holding back because the screen simply is not good enough for release.

I could have told them that at the beginning.

Because I know of no Liquid Crystal Display which is of any use for anything except a wrist watch. Those who still do not believe this lowly worm, please look at the incredibly expensive Hewlett Packard Lap Computer which has the best LCD display yet seen in captivity – but the display is still not much good.

I have the strongest doubts that Apple will ever release the pop up screen in Liquid Crystal Display form. □

BACK UP YOUR DISKS

ESSENTIAL DATA DUPLICATOR

EDD is the most powerful disk duplicator available for your Apple™ computer. Unlike the copycards, which only copy single load programs, EDD backs up your entire disk. EDD can back up more protected software than all other copy programs or copycards put together. Since EDD is automatic, you will no longer have to change parameters to duplicate most disks, although every parameter is fully documented in our extensive manual. We also provide updated EDD program lists.

$98.95

UTILITY SOFTWARE
83 Hall Street, Bondi Beach NSW 2026.
Telephone (02) 302105.

Was Rembrandt really an Australian?

It's easy to see why one American reviewer got our all-Australian software confused with a European masterpiece. Our programs have been praised in England and pirated in Taiwan. In Australia, they've been used and used and used. We want to make more all-Australian programs that are solidly useful.

The success we've had is enabling us to do just that. We have been forming a team of talented programmers whose sole aim is quality. We're preparing a series of masterpieces. None of them will be paintings, all of them will be the finest programs for the finest computers. They will all bear the name Zardax, as your guarantee of quality.

What's all this got to do with Rembrandt? Read on.

— *"this word processor is a piece of cake to use." Infoworld 22/2/82*
— *"superbly easy to use, quick, convenient." PCAL Report Dec 83*
— *"in many ways better than a dedicated word processor." The Australian 31/3/84*
— *"if word processing is an art, Zardax is a Rembrandt." In-Cider March 83*
— *"highly recommended." Softalk Jan 84*
— *"Zardax wins praise for being well-nigh bulletproof." Your Computer April 84*

Zardax®

Zardax Word Processor I	— for Dos. 3.3 on the Apple II computer
Zardax Word Processor II	— for ProDOS on the Apple //e and //c
Zardax Word Processor III	— for the Apple /// computer
Zardax WRITER	— a new generation program for the IBM PC

COMPUTER SOLUTIONS

16 Carrara Street,
Mt Gravatt Q. 4122
P.O. Box 397,
Mt Gravatt Q. 4122
(07) 349 9883

Publishers of Zardax software

Introducing a Teacher You Can Give to Your Apple.

For the first time in Australia, quality educational courseware is available for your Apple IIE or your Apple II Plus. Control Data PLATO® is proud to announce that from December 1st, new lessons from the vast PLATO computer based education library will be available for your Apple micro.

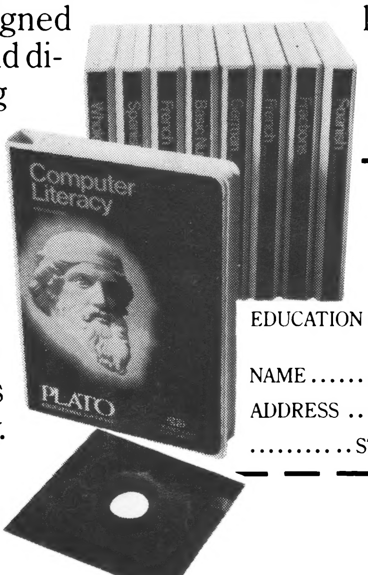
Control Data PLATO computer-based education, recognised as a world leader in educational software, is a highly interactive system of self-paced, one-to-one instruction. Developed over 20 years, it has been thoroughly tested and proven in business, industry, and the academic community. PLATO lessons are designed and programmed to motivate and direct students as well as helping them develop a sense of accomplishment and personal progress.

For example, the course titled Computer Literacy provides the foundation for simple programming as well as giving a brief introduction to the uses of computers in today's society. There are lessons in building vocabulary in French, German and Spanish as well as lessons in Physics, Fractions, Decimals and Basic Number Facts.

Today, more than ever before, all members of society need to learn. PLATO computer-based education is designed and paced for the individual. This means easier learning with infinite patience and personalization.

As traditional education methods become progressively more costly and less efficient you owe it to yourself to bring the power of PLATO learning to your Apple.

If you would like to be on our mailing list for future information on PLATO Courseware for your Apple, please fill in and mail the coupon below.



PLATO

Changing How The World Learns.

Mail to: Marketing Manager,
EDUCATION SERVICES, CONTROL DATA AUSTRALIA PTY. LTD.,
493 St. Kilda Road, Melbourne 3004.

NAME....................................

ADDRESS....................................

..........STATE....... POSTCODE.......

CONTROL DATA

Burrows Doble Lawrence CDA/616